Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1968-06

Determination of network functions by topological methods.

Chang, Wen-tao

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/40045

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun

DUDLEY KNOX LIBRARY

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

UNITED STATES
NAVAL POSTGRADUATE SCHOOL

THESIS

DETERMINATION OF NETWORK
FUNCTIONS BY TOPOLOGICAL METHOD

by

Wen-Tao Chang

June 1968

Thesis
C372

DETERMINATION OF NETWORK FUNCTIONS BY TOPOLOGICAL METHOD

by

Wen-Tao Chang
Lieutenant Commander, Chinese Navy
B.S., Naval Academy, Taiwan, ROC, 1956 and
B.E., Naval Institute of Technology, Taiwan, ROC, 1964

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1968

Signature of Author Wen-Tao Chang

Approved by Shu-gar Chan
Thesis Advisor

Charles H Rothauge
Chairman, Department of Electrical Engineering

W. F. Koehler for R. F. Rinehart
Academic Dean

Thesis C372 c.2

ABSTRACT

Two new methods, suitable for either hand or digital computer calculations, are developed for the determination of all the trees of a nonoriented linear graph. Computer programs based on these methods are written. They greatly reduce the computation time while minimizing computer storage requirements.

In addition, a program for the determination of network functions is also developed by using topological relationships. This program is sufficiently generalized to permit its use for either a one port or a two port passive network without mutual inductances.

TABLE OF CONTENTS

Section Page

1. Introduction 9

2. Topological Formulas 10

3. Tree Finding Methods 34

4. Computer Program for the Determination of Network Functions 65

5. Conclusion 72

6. Bibliography 73

Appendix

I. Tree Finding Computer Program I by Dichotomy Method 75

II. Tree Finding Computer Program II by T-Triangle Method 89

III. Computer Program for Network Functions 107

LIST OF ILLUSTRATION

Figure		Page
2-1	1-P Network	15
2-2	1-P Network Excited by a Current Source	15
2-3	1-P Network Excited by a Voltage Source	15
2-4	1-P Network for Illustrating Driving-point Admittance	16
2-5	Network, Illustrating Example 2.1	17
2-6	Graph, Illustrating Example 2.1	17
2-7	Graph, Illustrating Example 2.1	18
2-8	2-P Network	22
2-9	2-P Network for Illustrating Ratio Transfer Functions	27
2-10	Network, Illustrating Example 2.3	29
3-1	Graph, Illustrating Example 3.1	37
3-2	Graph, Illustrating Example 3.2	38
3-3	Input Data Cards of Program I	42
3-4	Graphs, Illustrating T-triangles	46
3-5	Graph, Illustrating Theorem 3.5	50
3-6	Graph, Illustrating Theorem 3.6	51
3-7	Graph, Illustrating the Algorithm of T-triangle Method	53
3-8	Graph, Illustrating the Algorithm of T-triangle Method	53
3-9	Graph, Illustrating the Algorithm of T-triangle Method	53
3-10	Schematic Diagram, Illustrating the Algorithm of T-triangle Method	54
3-11	Graph, Illustrating Example 3.3	55
3-12	Graph, Illustrating Example 3.4	55
3-13	Graph, Illustrating Example 3.5	57

Figure		Page
3-14	Input Data Cards of Program II	62
3-15	Graph, Illustrating 'Link Triangle'	63
4-1	One-port and Two-port Networks	66

ACKNOWLEDGEMENT

The author wishes to thank Professor Shu-Gar Chan for his guidance during the period of research and the preparation of this thesis.

1. Introduction.

Topological formulas for analyzing electrical networks have become popular in recent years. The main reasons for the current interest in this field are that they are relatively simple to manipulate when compared with the conventional method and that topological formulas readily implemented by means of computer programming. The topological formulas for analyzing an electrical network and for computing network functions have been studied by a number of investigators [1]-[8]. In section 2, topological formulas for evaluating various network functions of a passive network without mutual inductances are briefly discussed.

Trees of a network play an important role in topological network analysis. How all the trees of a network can be found effectively and economically is the main task with which this thesis is concerned. In this paper two new tree-finding methods, which are named the 'dichotomy method' and the 'T-triangle method', have been developed. The main advantages of these two methods are that there is no duplication, computation time is reduced, and they are suitable for both hand calculation and computer programming. The algorithm and computer program for each method are included in section 3.

A computer program for the determination of network functions is presented in section 4. The topological formulas, modified by Chan and Chan [6] and [8], which permit the determination of network functions by means of only a single tree finding subroutine, were used in this program. This program can handle a rather complex network due to the smaller memory requirement achieved by the 'T-triangle' tree finding method.

2. Topological Formulas.

2.1 General Description.

The basic laws used to solve for the currents and voltages in an electrical network are Kirchhoff's current law and Kirchhoff's voltage law. By using these laws one can write a set of loop (mesh) equations or a set of node equations from a network. The ordinary method of solving a set of equations to obtain a desired network function from the ratio of a determinant and its cofactor is cumbersome. Since the information in the determinant contains a great deal of redundancy (i.e., in the expansion of the determinant many terms occur more than once, but with opposite signs), its evaluation will be lengthy due to cancellations. As we see from the properties of an incidence matrix and circuit matrix, [1], [2], the determinant and cofactor of a system equations can be evaluated directly from the trees, cotrees and K-trees of the network. The topological analysis of electrical networks, therefore, provides a shortcut in evaluating network determinants and cofactors because the usual cancellations inherent in the evaluation of the network determinants are avoided. Moreover, it displays in a very intuitive manner, the causal relationships between the several variables under study.

In network synthesis, trees of a network will give the important information that significantly affects the network functions. Consequently, trees of a network play an important role in topological analysis and synthesis.

2.2 Definition of Terms and Symbols.

Edge or element - a line segment together with its end points.

Vertex or node - an end point of an edge.

Graph - a finite collection of edges such that no two edges have a point in common that is not a vertex. This is also called a linear graph.

Subgraph - a subset of edges of the graph.

Connected graph - a graph G is connected if there exists a path between any two vertexes of the graph.

Circuit or loop - a subgraph in which every vertex has two edges incident to it.

Tree - a connected subgraph of a connected graph containing all the vertices of the graph but containing no circuit.

Cotree - a set of chords, which includes all the edges of the complement of a tree in a graph G.

2-tree - a subgraph of a tree of G formed by removing any one edge from the tree. In other words, it is a pair of unconnected circuitless subgraphs which together include all the vertices of the graph.

3-tree - three unconnected circuitless subgraphs obtained by removing any one edge from a 2-tree of the graph.

Incidence matrix, A_a - a matrix with rows corresponding to vertices and columns to edges of a graph, and the entry

a_{ij} = 1 if the edge j is incident at vertex i

a_{ij} = 0 if the edge j is not incident at vertex i

Circuit matrix, B_a - a matrix with one row for each circuit in the graph, and the entry

b_{ij} = 1 if the edge j is in circuit i

b_{ij} = 0 if the edge j is not in circuit i.

A = incidence matrix with any one dependent row deleted

B = circuit matrix with the dependent rows deleted.

2.3 Concept of Topology in Electrical Networks

Kirchhoff's Laws

The fundamental equations of electrical network theory are:

(a) Kirchhoff's current law - the sum of the instantaneous currents out of any node are zero; in matrix notation it is

$$A\ I_e(T) = 0 \tag{2-3-1}$$

where A is the incidence matrix of a network as defined in section 2.2 and $I_e(t)$ is a column matrix of branch current.

(b) Kirchhoff's voltage law - the sum of instantaneous branch voltages around any loop is zero; in matrix notation it is

$$B\ V_e(T) = 0 \tag{2-3-2}$$

where B is the circuit matrix of a network as defined in section 2.2 and $V_e(t)$ is a column matrix of branch voltage, i.e.,

$$V_e(t) = L_e \frac{d}{dt} I_e(t) + R_e I_e(t) + C_e^{-1} \int_0^t I_e(t)dt + V_c(0^+) + E_e(t)$$

and where

L_e = element inductance matrix

R_e = element resistance matrix

C_e = element capacitance matrix

$E_e(t)$ = column matrix of branch voltage sources

The Laplace transformations of the fundamental equations are:

$$A\ I_e(S) = 0 \qquad \text{(KCL)} \tag{2-3-3}$$

$$B\,V_e(S) = 0 \qquad \text{(KVL)} \qquad (2\text{-}3\text{-}4)$$

where $V_e(s) = E_e(s) + Z_e(s) I_e(s) - L_e I_e(0^+) + \frac{1}{s} V_c(0^+)$

and Z_e is the element impedance matrix in square form. When we discuss network functions the initial conditions are always set to zero, therefore

$$V_e(S) = E_e(S) + Z_e(S)\,I_e(S) \qquad (2\text{-}3\text{-}5)$$

and
$$I_e(S) = B'\,I_m(S) \qquad (2\text{-}3\text{-}6)$$

$$V_e(S) = A'\,V_n(S) \qquad (2\text{-}3\text{-}7)$$

where A' and B' are the transport of A and B respectively; I_m and V_n both are column matrices. The subscripts m and n imply mesh (loop) and vertex (node) respectively. Substituting (2-3-5) into (2-3-4) we have

$$B\left[E_e(s) + Z_e(s) I_e(s)\right] = 0 \qquad (2\text{-}3\text{-}8)$$

Substituting (2-3-6) into (2-3-8) we have

$$B\,Z_e(s)\,B'\,I_m(s) = -B\,E_e(s) \qquad (2\text{-}3\text{-}9)$$

Referring to equations (2-3-5) once more, solving for $I_e(s)$ and using $Z_e^{-1}(s) = Y_e(S)$, we obtain

$$I_e(s) = Y_e(s)\left[V_e(s) - E_e(s)\right] \qquad (2\text{-}3\text{-}10)$$

where Y_e is the element admittance matrix in square form. Then substitute (2-3-10) and (2-3-7) into (2-3-3) which becomes

$$A\,Y_e(s)\,A'\,V_n(s) = A\,Y_e(s)\,E_e(s) \qquad (2\text{-}3\text{-}11)$$

Equation (2-3-9) can be written in the form:

$$Z_m(s)\,I_m(s) = E'_m(s) \qquad (2\text{-}3\text{-}12)$$

where $Z_m(s) \equiv B\, Z_e(s)\, B'$, the mesh impedance matrix and E_m is the column matrix of voltage sources around a loop. The polarities of the voltage sources are chosen to be positive in the direction of the loop current.

Equation (2-3-11) can be written in the form

$$Y_n(S)\, V_n(S) = I_n(S) \tag{2-3-13}$$

Where $Y_n(S) \equiv A\, Y_e A'$, the node admittance matrix and I_n represents the node current sources in column matrix form.

Equation (2-3-12) is known as the loop (mesh) equations in matrix form and equation (2-3-13) is known as the node equation in matrix form.

The utilization of the properties of incidence matrices and circuit matrices and the development of topological formulas to evaluate the mesh impedance and node admittance determinants and their cofactors by using tree, cotree and 2-tree concepts have been discussed by Mayeda [1], Weinberg [2], Chan [3], Reza [4], and Seshu and Reed [5]. Therefore, the formulas are merely outlined here.

(1) $$\Delta = |Y_n(s)| = \sum \text{Tree admittance products} = V(Y) \tag{2-3-14}$$

i.e., the sum of all tree admittance products of a network, N. The notation Δ is the node determinant.

(2) $$\Delta_{ii} = |A_{-i}\, Y_e(s)\, A'_{-i}| = \sum T_{2_{i,r}} \text{ admittance products} = W_{i,r}(Y) \tag{2-3-15}$$

i.e., the sum of all the 2-tree admittance products of a network, N, with each 2-tree formed in two subgraphs; one subgraph containing node i and the other subgraph containing the reference node r. This is identical to the sum of all tree admittance products of the graph obtained by identifying the i^{th} node with the reference node. This graph is designated $N_{i,r}$. The notation Δ_{ii} is the (i,i) cofactor of Δ.

(3) $\Delta = |Z_m(s)| = \sum$ cotree impedance products (2-3-16)

i.e., sum of all the chord set (complement edges of a tree) impedance products of a network, N. The notation Δ is the mesh determinant.

(4) $\Delta_{ii} = |B_{-i} Z_e(s) B'_{-i}| = \sum$ cotree impedance products (2-3-17)

of $N_1 = C[V_1(z)]$

where N_1 is the network obtained by deleting element y_j which is an element in loop i but not in any other loop. The notation Δ_{ii} is the (i,i) cofactor of Δ.

2.4 One-port Networks

One port network (1-p network) is a network with one pair of nodes designated as "input terminal nodes," i.e., the driving point terminals. A one port network is shown in Fig. 2-1.

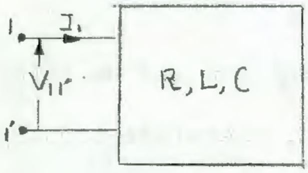


Fig. 2-1

1-p network

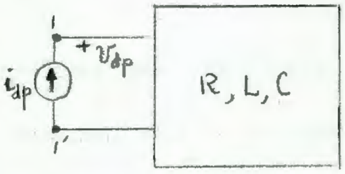


Fig. 2-2

1-p network, excited by current source

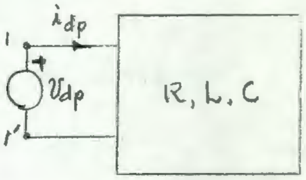


Fig. 2-3

1-p network excited by voltage source

The network can be excited across input terminals 1-1' by either a current source, Fig. 2-2, or by a voltage source, Fig. 2-3.

Driving Point Admittance

The driving point admittance is defined as

$$Y_{dp}(s) = \frac{I_{dp}(S)}{V_{dp}(S)}\Bigg|_{\text{initial condition}=0} = \frac{I_1}{V_{11'}}$$

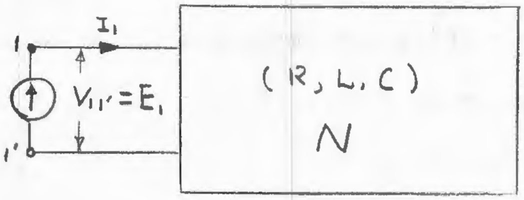


Fig. 2-4

1 - p network with node 1' as the reference node.

Starting from the node equation for a 1-p network with v-1 independent nodes, from (2-3-13)

$$Y_n(S)\; V_n(S) = I_n(S) \tag{2-4-1}$$

or in detail

$$\begin{bmatrix} Y_{11} & Y_{12} & \cdots & Y_{1,v-1} \\ Y_{21} & Y_{22} & \cdots & Y_{2,v-1} \\ \cdot & \cdot & \cdots & \cdot \\ Y_{v-1,1} & Y_{v-1,2} & \cdots & Y_{v-1,v-1} \end{bmatrix} \begin{bmatrix} V_{11'} \\ V_{21'} \\ \vdots \\ V_{v-1,1'} \end{bmatrix} = \begin{bmatrix} I_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2-4-2}$$

The solution for $V_{11'}$ is

$$V_{11'} = \frac{\Delta_{11'}}{\Delta}\; I_1 \tag{2-4-3}$$

then we get

$$Y_{dp}(s) = \frac{\Delta}{\Delta_{11'}} \tag{2-4-4}$$

Since Δ and $\Delta_{11'}$ are the determinant and its cofactor of the node admittance matrix of the network, N, therefore topological formulas (2-3-14) and (2-3-15) can be applied here, and the desired topological formula for the driving point admittance is

$$Y_{dp} = \frac{V(Y)}{W_{1,1'}(Y)} \tag{2-4-5}$$

$$\text{or} \quad Y_{dp}(S) = \frac{\sum \text{tree admittance products of network } \boxed{N}}{\sum \text{tree admittance products of network } \boxed{N_{1,1'}}} \tag{2-4-6}$$

Example 2.1. A typical network is shown in Fig. 2-5.

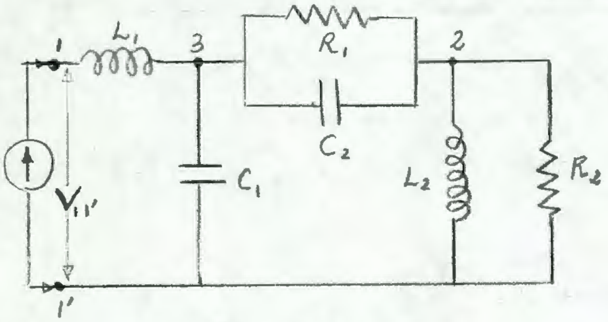


Fig 2-5

1-P network, N.

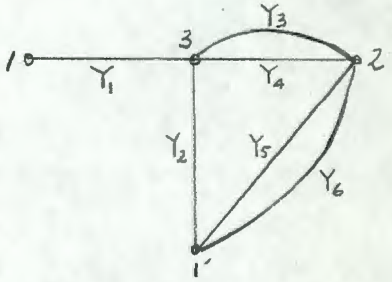


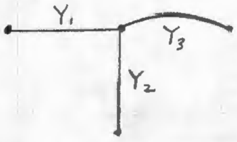
Fig 2-6

The graph of network N.

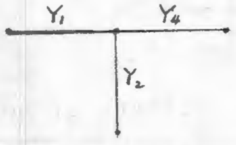
Table 1. The given values of parameters

Parameter	$L_1 = 1$	$C_1 = 2$	$R_1 = 1$	$C_2 = 3$	$L_2 = 1$	$R_2 = \frac{1}{2}$
Admittance	$Y_1 = \frac{1}{s}$	$Y_2 = 2s$	$Y_3 = 1$	$Y_4 = 3s$	$Y_5 = \frac{1}{s}$	$Y_6 = 2$
Impedance	$Z_1 = s$	$Z_2 = \frac{1}{2s}$	$Z_3 = 1$	$Z_4 = \frac{1}{3s}$	$Z_5 = s$	$Z_6 = \frac{1}{2}$

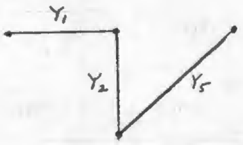
Trees of graph N:



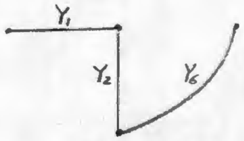
$Y_1Y_2Y_3 = 2$



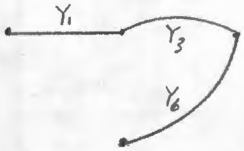
$Y_1Y_2Y_4 = 6s$



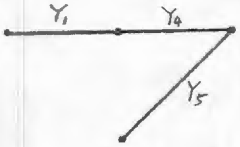
$Y_1Y_2Y_5 = \frac{2}{s}$



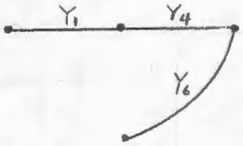
$Y_1Y_2Y_6 = 4$



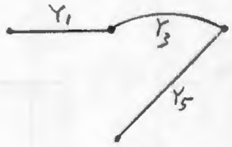
$Y_1Y_3Y_6 = \frac{2}{s}$



$Y_1Y_4Y_5 = \frac{3}{s}$



$Y_1Y_4Y_6 = 6$



$Y_1Y_3Y_5 = \frac{1}{s^2}$

$$\therefore V(Y) = \frac{1}{s^2} + \frac{7}{s} + 12 + 6s$$

To get $W_{1,1'}(Y)$, we short nodes 1 - 1' then we get $N_{1,1'}$ shown in Fig. 2-7

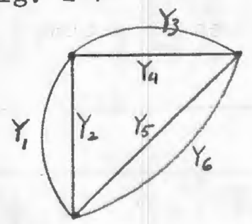
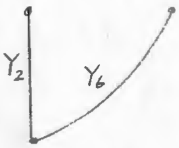
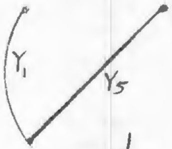


Fig. 2-7. The graph of $N_{1,1'}$.

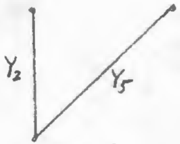
Trees of graph $N_{1,1'}$ are



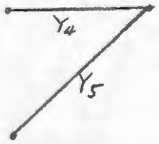
$Y_2Y_6 = 4s$



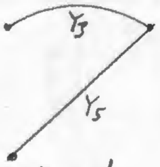
$Y_1Y_5 = \frac{1}{s^2}$



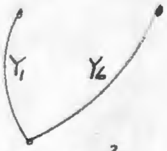
$Y_2Y_5 = 2$



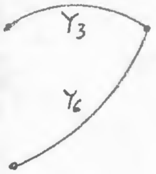
$Y_4Y_5 = 3$



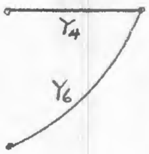
$Y_3Y_5 = \frac{1}{s}$



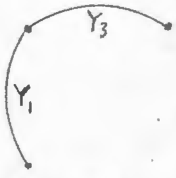
$Y_1Y_6 = \frac{2}{s}$



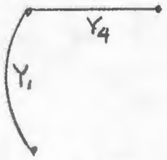
$Y_3Y_6 = 2$



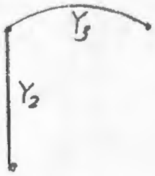
$Y_4Y_6 = 6s$



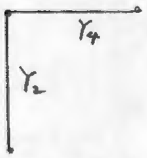
$Y_1Y_3 = \frac{1}{s}$



$Y_1Y_4 = 3$



$Y_2Y_3 = 2s$



$Y_2Y_4 = 6s^2$

$$\therefore \; W_{1,1'}(Y) = \frac{1}{s^2} + \frac{4}{s} + 10 + 12s + 6s^2 \tag{2-4-7}$$

Therefore

$$Y_{dp} = \frac{\frac{1}{s^2} + \frac{7}{s} + 12 + 6s}{\frac{1}{s^2} + \frac{4}{s} + 10 + 12s + 6s^2} \tag{2-4-8}$$

$$Y_{dp} = \frac{1 + 7s + 12s^2 + 6s^3}{1 + 4s + 10s^2 + 12s^3 + 6s^4} \tag{2-4-9}$$

Driving Point Impedance

The driving point impedance of a 1-p network N, see Fig. 2-4, is defined as

$$Z_{dp}(s) = \frac{E_1(s)}{I_1(s)} \quad \text{all initial values} = 0 \tag{2-4-10}$$

From equation (2-3-12), the loop equation of 1-p network N with ℓ independent loops is

$$\begin{bmatrix} Z_{11} & Z_{12} & \cdots & Z_{1\ell} \\ Z_{21} & Z_{22} & \cdots & Z_{2\ell} \\ \cdots & \cdots & \cdots & \cdots \\ Z_{\ell 1} & Z_{\ell 2} & \cdots & Z_{\ell\ell} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_\ell \end{bmatrix} = \begin{bmatrix} E_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2-4-11}$$

where Z_{jk} = the impedance in S-domain common to loop j and k

I_j = loop current around loop j

E_j = net voltage rise around loop j

Solve for I_1

$$I_1 = \frac{\Delta_{11}}{\Delta} E_1 \tag{2-4-12}$$

so that

$$Z_{dp} = \frac{\Delta}{\Delta_{11}} \tag{2-4-13}$$

Z_m (mesh impedance matrix) in equation (2-3-12) is not just for the network N, but in addition, includes one loop which consists of the driving source, i.e., the loop established by the first row in equation

(2-3-12). Z_m for the network alone is the submatrix of Z_m obtained by deleting the first row and first column. Therefore, with a little modification to the topological formulas (2-3-16) and (2-3-17), [1], we have

$$\Delta_{11} = C[V(Z)] \tag{2-4-14}$$

$$\Delta = C[W_{1,1'}(Z)] \tag{2-4-15}$$

Hence the topological formula for the driving point impedance is

$$Z_{dp} = \frac{C[W_{1,1'}(Z)]}{C[V(Z)]} \tag{2-4-16}$$

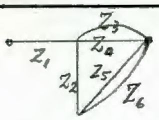
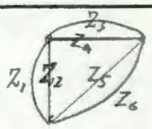
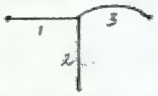
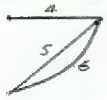
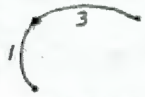
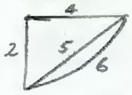
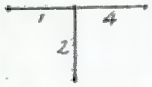
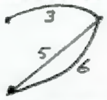
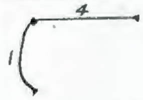
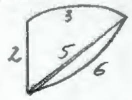
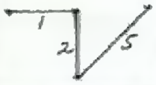
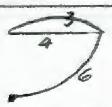
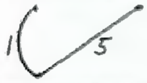
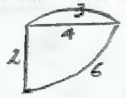
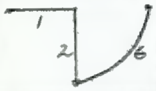
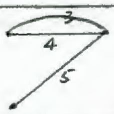
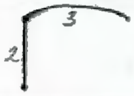
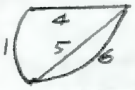
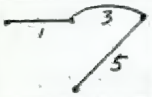
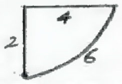
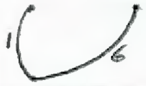
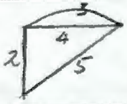
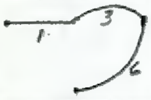
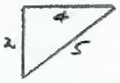
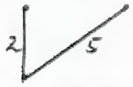
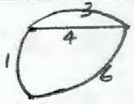
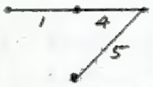
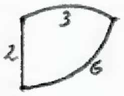
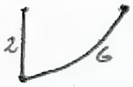
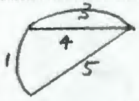
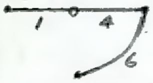
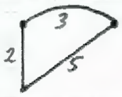
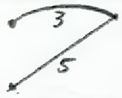
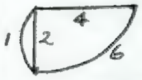
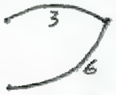
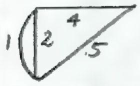
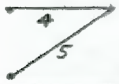
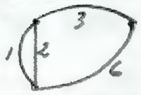
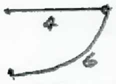
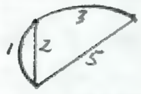
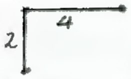
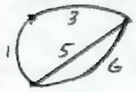
To avoid the cumbersome manipulation on cotrees, we can compute Z_{dp} from Y_{dp} :

$$Z_{dp} = \frac{E_1}{I_1} = \frac{1}{I_1/E_1} = \frac{1}{Y_{dp}} \tag{2-4-17}$$

By the topological formula of Y_{dp} , equation (2-4-18), we have the more useful topological formula for Z_{dp} , thus

$$Z_{dp} = \frac{W_{1,1'}(Y)}{V(Y)} \tag{2-4-18}$$

Example 2.2. For the purpose of comparing results and in order to provide the same input for a computer program, let's consider the same 1-p network as in example 2.1. Also the given values of parameters are the same as in example 2.1.

Graph of network N: 			Graph of network $N_{1,1'}$ 		
Trees	Cotrees	Cotree Prod.	Trees	Cotrees	Cotree Prod.
		$Z_4Z_5Z_6=\frac{1}{6}$			$Z_2Z_4Z_5Z_6=\frac{1}{12s}$
		$Z_3Z_5Z_6=\frac{s}{2}$			$Z_2Z_3Z_5Z_6=\frac{1}{4}$
		$Z_3Z_4Z_6=\frac{1}{6s}$			$Z_2Z_3Z_4Z_6=\frac{1}{12s^2}$
		$Z_3Z_4Z_5=\frac{1}{3}$			$Z_1Z_4Z_5Z_6=\frac{s}{6}$
		$Z_2Z_4Z_6=\frac{1}{12s^2}$			$Z_2Z_3Z_4Z_5=\frac{1}{6s}$
		$Z_2Z_4Z_5=\frac{1}{6s}$			$Z_1Z_3Z_4Z_6=\frac{s^2}{2}$
		$Z_2Z_3Z_6=\frac{1}{4s}$			$Z_1Z_3Z_4Z_5=\frac{s}{3}$
		$Z_2Z_3Z_5=\frac{1}{2}$			$Z_1Z_2Z_4Z_6=\frac{1}{12s}$
					$Z_1Z_2Z_4Z_5=\frac{1}{6}$
					$Z_1Z_2Z_3Z_6=\frac{1}{4}$
					$Z_1Z_2Z_3Z_5=\frac{s}{2}$
					$Z_1Z_3Z_5Z_6=\frac{1}{6}$
$C[V(z)]=\frac{1}{12}s^{-2}+\frac{7}{12}s^{-1}+1+\frac{1}{2}s$			$C[W_{1,1'}(z)]=\frac{1}{12}s^{-2}+\frac{4}{12}s^{-1}+\frac{10}{12}+s+\frac{1}{2}s^2$		

$$Z_{dp} = \frac{\frac{1}{12} s^{-2} + \frac{4}{12} s^{-1} + \frac{10}{12} + s + \frac{1}{2} s^{2}}{\frac{1}{12} s^{-2} + \frac{7}{12} s^{-1} + 1 + s} \qquad (2\text{-}4\text{-}19)$$

$$Z_{dp} = \frac{1 + 4s + 10s^{2} + 12s^{3} + 6s^{4}}{1 + 7s + 12s^{2} + 6s^{3}} \qquad (2\text{-}4\text{-}20)$$

If we use equation (2-4-17) in conjunction with the value of Y_{dp} obtained in example 2.1, then we have

$$Z_{dp} = \frac{1}{Y_{dp}} = \frac{1 + 4s + 10s^{2} + 12s^{3} + 6s^{4}}{1 + 7s + 12s^{2} + 6s^{3}} \qquad (2\text{-}4\text{-}21)$$

Notice this result agrees with the previous one.

2.5 Two-port Networks

A network with one pair of nodes (1,1') designated as the input terminals and one other pair of nodes (2,2') designated as the output terminals is called a two port network (2-P). It is shown in Fig. 2-8.

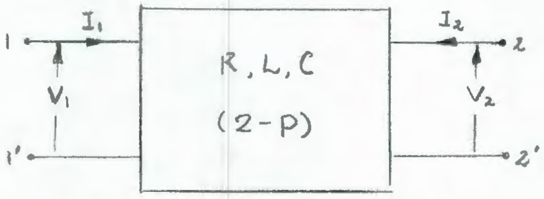


Fig. 2-8. A two-port network, with nodes (1,1') and (2,2') designated as the input and output terminals respectively.

There are four variables to be related in a two-port network, i.e. I_1, the input current; V_1, the input voltage; I_2, the output current; and V_2, the output voltage. The topolotical formulas associated to the ratios of the different variables of a 2-p network will be discussed in the following sections.

Open Circuit (O.C.) Impedance Function

From Fig. 2-8, we can write the loop equations as

$$\begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} Z_{11}(s) & Z_{12}(s) \\ Z_{21}(s) & Z_{22}(s) \end{bmatrix} \begin{bmatrix} I_1(s) \\ I_2(s) \end{bmatrix} \qquad (2\text{-}5\text{-}1)$$

The parameters Z_{11}, Z_{12}, Z_{21}, and Z_{22} are called the open circuit impedance functions of the passive two-port network. They are denoted as Z_{oc}, the open circuit impedance matrix, thus

$$Z_{oc} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \qquad (2\text{-}5\text{-}2)$$

From Eq. (2-5-1), clearly we can evaluate these functions in the following manner:

$$\begin{aligned} Z_{11} &= \left.\frac{V_1}{I_1}\right|_{\substack{I_2=0 \\ (\text{open } 2\text{-}2')}} & Z_{12} &= \left.\frac{V_1}{I_2}\right|_{\substack{I_1=0 \\ (\text{open } 1\text{-}1')}} \\ Z_{21} &= \left.\frac{V_2}{I_1}\right|_{\substack{I_2=0 \\ (\text{open } 2\text{-}2')}} & Z_{22} &= \left.\frac{V_2}{I_2}\right|_{\substack{I_1=0 \\ (\text{open } 1\text{-}1')}} \end{aligned} \qquad (2\text{-}5\text{-}3)$$

Notice for a reciprocal network we have

$$Z_{12} = Z_{21} \qquad (2\text{-}5\text{-}4)$$

Let node 1' be the reference node in Fig. 2-8, and assume there are n independent nodes in the 2-p network, then the node equations are

$$\begin{aligned} Y_{11}V_1 + Y_{12}V_2 + \cdots\cdots Y_{1n}V_n &= I_1 \\ Y_{21}V_1 + Y_{22}V_2 + \cdots\cdots Y_{2n}V_n &= I_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ Y_{n1}V_1 + Y_{n2}V_2 + \cdots\cdots Y_{nn}V_n &= I_n \end{aligned} \qquad (2\text{-}5\text{-}5)$$

Solving the above equation, [3] [5] we get

$$Z_{oc} = \frac{1}{\Delta}\begin{bmatrix} \Delta_{11} & \Delta_{12} - \Delta_{12'} \\ \Delta_{12} - \Delta_{12'} & \Delta_{22} + \Delta_{2'2'} - \Delta_{22'} - \Delta_{2'2} \end{bmatrix} \qquad (2\text{-}5\text{-}6)$$

where Δ and Δ_{ij} are the determinant and its cofactor of the node-admittance matrix of Eq. (2-5-5).

From the topological formulas stated in Section 2.3, we have

$$\begin{aligned} \Delta_{12} - \Delta_{12'} &= W_{12,1'} - W_{12',1'} \\ &= (W_{12,1'2'} + W_{122',1'}) - (W_{12',12} + W_{122',1'}) \\ &= W_{12,1'2'} - W_{12',1'2} \qquad (2\text{-}5\text{-}7) \end{aligned}$$

and

$$\begin{aligned} \Delta_{22} + \Delta_{2'2'} - \Delta_{22'} - \Delta_{2'2} &= W_{2,1'} + W_{2',1'} - W_{22',1'} - W_{2'2,1'} \\ &= W_{2,1'2'} + W_{22',1'} + W_{2',21'} + W_{22',1'} - W_{22',1'} - W_{2'2,1'} \\ &= W_{2,1'2'} + W_{2',1'2} \\ &= W_{2,2'} \qquad (2\text{-}5\text{-}8) \end{aligned}$$

Therefore, the topological formulas for open circuit impedance functions are

$$Z_{oc} = \frac{1}{V(Y)}\begin{bmatrix} W_{1,1'} & W_{12,1'2'} - W_{12',1'2} \\ W_{12,1'2'} - W_{12',1'2} & W_{2,2'} \end{bmatrix} \qquad (2\text{-}5\text{-}9)$$

However, the more useful topological formulas of Z_{oc} for digital computation are developed by Chan and Chan, [6][8], that is

$$Z_{oc} = \frac{1}{V(Y)}\begin{bmatrix} W_{1,1'} & \frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'}) \\ \frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'}) & W_{2,2'} \end{bmatrix} \qquad (2\text{-}5\text{-}10)$$

Short Circuit (S.C.) Admittance Functions

If we solve Eq. (2-5-1) for I in terms of Z_{oc}, V_1 and V_2 then we have

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \left[Z_{oc} \right]^{-1} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \frac{1}{\Delta_Z} \begin{bmatrix} Z_{22} & -Z_{12} \\ -Z_{21} & Z_{11} \end{bmatrix} \tag{2-5-11}$$

We define the short circuit admittance matrix Y_{SC} as

$$Y_{sc} = Z_{oc}^{-1} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \tag{2-5-12}$$

Then the entries of Y_{SC}, which we called the short circuit admittance functions of the passive 2-P, can be measured in the following manner:

$$\begin{aligned} Y_{11} &= \left.\frac{I_1}{V_1}\right|_{\substack{V_2=0 \\ (\text{short } 2-2')}} & \qquad Y_{12} &= \left.\frac{I_1}{V_2}\right|_{\substack{V_1=0 \\ (\text{short } 1-1')}} \\ Y_{21} &= \left.\frac{I_2}{V_1}\right|_{\substack{V_2=0 \\ (\text{short } 2-2')}} & \qquad Y_{22} &= \left.\frac{I_2}{V_2}\right|_{\substack{V_1=0 \\ (\text{short } 1-1')}} \end{aligned} \tag{2-5-13}$$

Also notice for a reciprocal network we have

$$Y_{12} = Y_{21} \tag{2-5-14}$$

The expressions for the short circuit admittance functions in terms of the cofactors of the node-admittance matrix are given [3][1] as

$$Y_{sc} = \frac{1}{\Delta_{1122} + \Delta_{112'2'} - 2\Delta_{1122'}} \begin{bmatrix} \Delta_{22} + \Delta_{2'2'} - 2\Delta_{22'} & \Delta_{12'} - \Delta_{12} \\ \Delta_{12'} - \Delta_{12} & \Delta_{11} \end{bmatrix} \tag{2-5-15}$$

where $\Delta_{ijk\ell}$ is the determinant obtained from Δ by deleting rows i and k and columns j and ℓ.

The denominator on the right hand side of Eq. (2-5-15) can be expressed in topological formula as [3][1]

$$\dot{\Delta}_{1112} + \dot{\Delta}_{112'2'} - 2\Delta_{1122'} = \sum U(Y)$$

$$= U_{12',2,1'} + U_{1,2,1'2'} + U_{12,2',1'} + U_{1,2',1'2} \qquad (2\text{-}5\text{-}16)$$

where $U_{12',2,1'}$ is the sum of 3-tree admittance products. The 3-tree designated by the indices of U results from the subgraphs with vertices 1,2' in one part, vertex 2 in another part and vertex 1' in the third part. Similar interpretations can be applied to the other terms in (2-5-16).

The topological expressions for those terms in the matrix of Eq. (2-5-15) are shown in the previous section. Consequently, the topological formula for short circuit admittance functions are

$$Y_{sc} = \frac{1}{\sum U}\begin{bmatrix} W_{2,2'} & W_{12',1'2} - W_{12,1'2'} \\ W_{12',1'2} - W_{12,1'2'} & W_{1,1'} \end{bmatrix} \qquad (2\text{-}5\text{-}17)$$

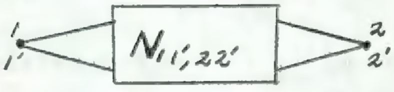
However the more useful topological formulas for Y_{SC} for digital computation are given by Chan and Chan [6] [8], that is

$$Y_{sc} \doteq \frac{1}{\sum U}\begin{bmatrix} W_{2,2'} & \frac{1}{2}(W_{1,2}+W_{1',2'}-W_{1,2'}-W_{2,1'}) \\ \frac{1}{2}(W_{1,2}+W_{1',2'}-W_{1,2'}-W_{2,1'}) & W_{1,1'} \end{bmatrix} \qquad (2\text{-}5\text{-}18)$$

It has been shown by Chan [7] that

$\sum U = \sum$ Tree admittance product of the network which is obtained by identifying nodes 1-1' and nodes 2-2'. (2-5-19)

i.e.,

$\sum U =$ V(Y) of the network  (2-5-20)

Voltage Ratio Transfer Functions

The voltage ratio transfer function, μ_{12}, of a 2-P Network (Fig. 2-9) is defined as the ratio of the output terminal voltage to the input terminal voltage when $I_2 = 0$.

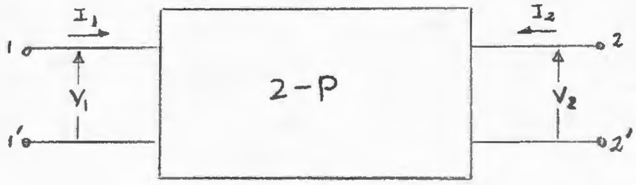


Fig. 2-9

Thus

$$\mu_{12} = \left.\frac{V_2}{V_1}\right|_{I_2=0} \quad (\text{open } 2\text{-}2') \tag{2-5-21}$$

Similarly

$$\mu_{21} = \left.\frac{V_1}{V_2}\right|_{I_1=0} \quad (\text{open } 1\text{-}1') \tag{2-5-22}$$

From Eq. (2-5-1) and Eq. (2-5-10) we obtain the following equations:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \frac{1}{V(Y)} \begin{bmatrix} W_{1,1'} & \frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1';2'}) \\ \frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1';2'}) & W_{2,2'} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} \tag{2-5-23}$$

when terminals 2-2' are open, $I_2 = 0$. Then,

$$V_1 = \frac{W_{1,1'}}{V(Y)} I_1 \tag{2-5-24}$$

$$V_2 = \frac{\frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1';2'})}{V(Y)} I_1 \tag{2-5-25}$$

Therefore,

$$\mu_{12} = \frac{V_2}{V_1}\bigg|_{I_2=0} = \frac{\frac{1}{2}\left(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'}\right)}{W_{1,1'}} \tag{2-5-26}$$

Similarly

$$\mu_{21} = \frac{V_1}{V_2}\bigg|_{I_1=0} = \frac{\frac{1}{2}\left(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'}\right)}{W_{2,2'}} \tag{2-5-27}$$

Current Ratio Transfer Functions.

The current ratio transfer function, α_{12} , of a 2-P Network (See Fig. 2-9) is defined as the ratio of the output to the input currents when the output terminals (2-2') are shorted, i.e., $V_2 = 0$. Thus

$$\alpha_{12} = \frac{I_2}{I_1}\bigg|_{V_2=0} \quad (\text{short } 2\text{-}2') \tag{2-5-28}$$

Similarly

$$\alpha_{21} = \frac{I_1}{I_2}\bigg|_{V_1=0} \quad (\text{short } 1\text{-}1') \tag{2-5-29}$$

From Eqs. (2-5-11) and (2-5-18) we have the following equations:

$$\begin{aligned}\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} &= \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \\ &= \frac{1}{\sum U} \begin{bmatrix} W_{2,2'} & \frac{1}{2}\left(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'}\right) \\ \frac{1}{2}\left(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'}\right) & W_{1,1'} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}\end{aligned} \tag{2-5-30}$$

By similar manipulations to those done for the voltage transfer function, we obtain

$$\alpha_{12} = \left.\frac{I_2}{I_1}\right|_{V_2=0} = \frac{\frac{1}{2}(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'})}{W_{2,2'}} \tag{2-5-31}$$

$$\alpha_{21} = \left.\frac{I_1}{I_2}\right|_{V_1=0} = \frac{\frac{1}{2}(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'})}{W_{1,1'}} \tag{2-5-32}$$

Example 2.3 Given a 2-p Network shown in Fig. 2-10

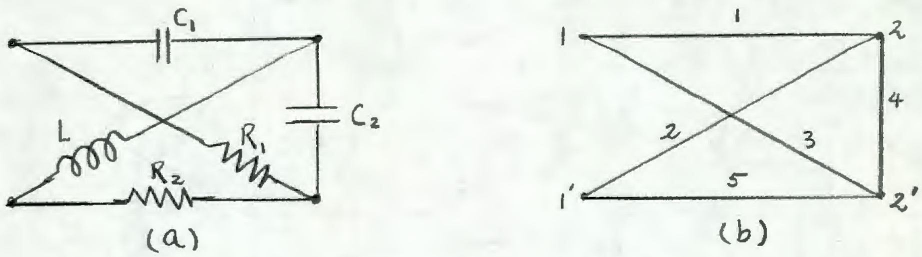


Fig. 2-10, (a) A given 2-P Network, (b) The graph corresponding to the Network shown in (a).

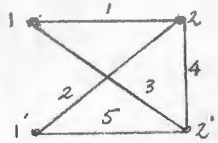
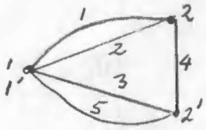
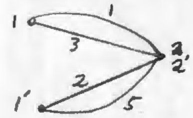
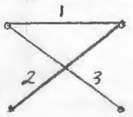
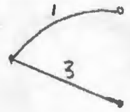
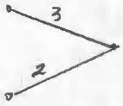
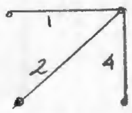
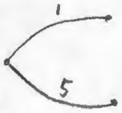
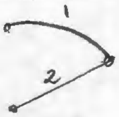
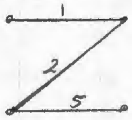
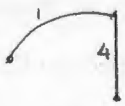
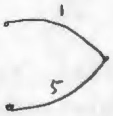
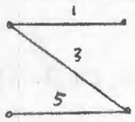
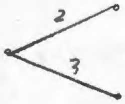
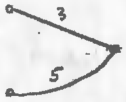
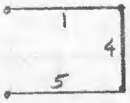
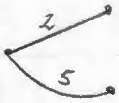
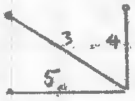
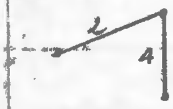
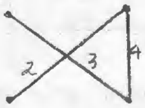
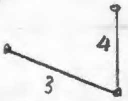
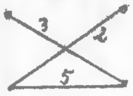
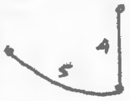
The values of the parameters in the Network are shown in Table 2.

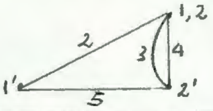
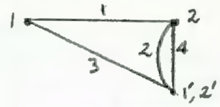
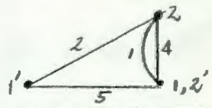
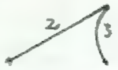
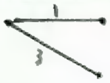
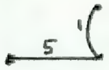
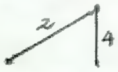
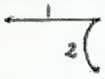
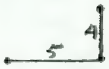
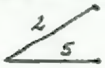
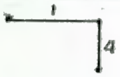
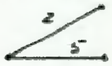
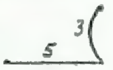
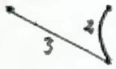
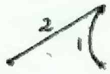
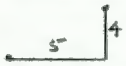
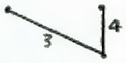
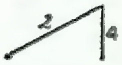
Table 2 Parameters of Example 2-3

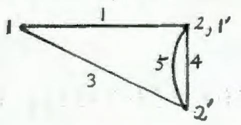
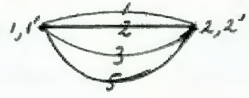
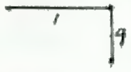
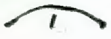
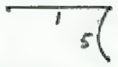
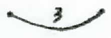
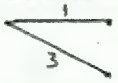
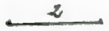
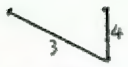
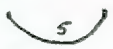
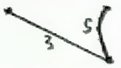
Edge	1	2	3	4	5
Parameter	C_1	L	R_1	C_2	R_2
Value	1.0 farad	0.5 henry	1.0 ohm	1.0 farad	1.0 ohm
Admittance	$Y_1 = 1.0s$	$Y_2 = \frac{2.0}{s}$	$Y_3 = 1.0$	$Y_4 = 1.0s$	$Y_5 = 1.0$

The network functions of the 2-P Network given above are obtained by the following three steps:

Step 1. List of all the trees, 2-trees, 3-trees and their admittance product.

Graph of network N 		Graph of $N_{1,1'}$ (short 1-1') 		Graph of $N_{2,2'}$ (short 2-2') 	
trees	tree products	2-trees	2-tree products	2-trees	2-tree products
	$Y_1Y_2Y_3 = 2$		$Y_1Y_3 = s$		$Y_2Y_3 = \frac{2}{s}$
	$Y_1Y_2Y_4 = 2s$		$Y_1Y_5 = s$		$Y_1Y_2 = 2$
	$Y_1Y_2Y_5 = 2$		$Y_1Y_4 = s^2$		$Y_1Y_5 = s$
	$Y_1Y_3Y_5 = s$		$Y_2Y_3 = \frac{2}{s}$		$Y_3Y_5 = 1$
	$Y_1Y_4Y_5 = s^2$		$Y_2Y_5 = \frac{2}{s}$		
	$Y_3Y_4Y_5 = s$		$Y_2Y_4 = 2$		
	$Y_2Y_3Y_4 = 2$		$Y_3Y_4 = s$		
	$Y_2Y_3Y_5 = \frac{2}{s}$		$Y_4Y_5 = s$		
$V(Y) = \frac{2}{s} + 6 + 4s + s^2$		$W_{1,1'}(Y) = \frac{4}{s} + 2 + 4s + s^2$		$W_{2,2'}(Y) = \frac{2}{s} + 3 + s$	

Graph of $N_{1,2}$ (short 1-2) 		Graph of $N_{1',2'}$ (short 1'-2') 		Graph of $N_{1,2'}$ (short 1-2') 	
2-trees	2-tree prod.	2-trees	2-tree prod.	2-trees	2-tree prod.
	$Y_2 Y_3 = \frac{2}{s}$		$Y_1 Y_3 = s$		$Y_1 Y_5 = s$
	$Y_2 Y_4 = 2$		$Y_1 Y_2 = 2$		$Y_4 Y_5 = s$
	$Y_2 Y_5 = \frac{2}{s}$		$Y_1 Y_4 = s^2$		$Y_2 Y_5 = \frac{2}{s}$
	$Y_3 Y_5 = 1$		$Y_2 Y_3 = \frac{2}{s}$		$Y_1 Y_2 = 2$
	$Y_4 Y_5 = s$		$Y_3 Y_4 = s$		$Y_2 Y_4 = 2$
$W_{1,2}(Y) = \frac{4}{s} + 3 + s$		$W_{1',2'}(Y) = \frac{2}{s} + 2 + 2s + s^2$		$W_{1,2'}(Y) = \frac{2}{s} + 4 + 2s$	

Graph of $N_{2,1'}$ (short 2-1') 		Graph of $N_{11',22'}$ (short 1-1' and 2-2') 	
2-trees	2-tree Prod.	2-trees	2-tree prod.
	$Y_1 Y_4 = s^2$		$Y_1 = s$
	$Y_1 Y_5 = s$		$Y_3 = \frac{2}{s}$
	$Y_1 Y_3 = s$		$Y_2 = 1$
	$Y_3 Y_4 = s$		$Y_5 = 1$
	$Y_3 Y_5 = 1$		
$W_{2,1'}(Y) = 1 + 3s + s^2$		$\Sigma U = \frac{2}{s} + 2 + s$	

Step 2

From the results of Step 1, we have

$$V(Y) = \frac{2}{s} + 6 + 4s + s^2$$

$$W_{1,1'}(Y) = \frac{4}{s} + 2 + 4s + s^2$$

$$W_{2,2'}(Y) = \frac{2}{s} + 3 + s$$

$$\frac{1}{2}(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'}) = \frac{1}{2}\left\{\left(\frac{4}{s} + 3 + s\right) + \left(\frac{2}{s} + 2 + 2s + s^2\right) - \left(\frac{2}{s} + 4 + 2s\right) - (1 + 3s + s^2)\right\}$$

$$= \frac{2}{s} - s$$

$$\frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'}) = s - \frac{2}{s}$$

$$\sum U = \frac{2}{s} + 2 + s$$

Step 3. By the topological formulas of (2-5-10), (2-5-18), (2-5-26), (2-5-27), (2-5-31) and (2-5-32), we obtain the following results.

(1) Open circuit impedance functions

$$Z_{11} = \frac{W_{1,1'}}{V(Y)} = \frac{\frac{4}{s} + 2 + 4s + s^2}{\frac{2}{s} + 6 + 4s + s^2} = \frac{4 + 2s + 4s^2 + s^3}{2 + 6s + 4s^2 + s^3} \qquad (2\text{-}5\text{-}33)$$

$$Z_{12} = Z_{21} = \frac{\frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'})}{V(Y)}$$

$$= \frac{s - \frac{2}{s}}{\frac{2}{s} + 6 + 4s + s^2} = \frac{s^2 - 2}{s^3 + 4s^2 + 6s + 2} \qquad (2\text{-}5\text{-}34)$$

$$Z_{22} = \frac{W_{2,2'}}{V(Y)} = \frac{\frac{2}{s} + 3 + s}{s^3 + 4s^2 + 6s + 2} = \frac{s^2 + 3s + 2}{s^3 + 4s^2 + 6s + 2} \tag{2-5-35}$$

(2) Short circuit admittance functions

$$Y_{11} = \frac{W_{2,2'}}{\Sigma U} = \frac{\frac{2}{s} + 3 + s}{\frac{2}{s} + 2 + s} = \frac{s^2 + 3s + 2}{s^2 + 2s + 2} \tag{2-5-36}$$

$$Y_{12} = Y_{21} = \frac{\frac{1}{2}(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'})}{\Sigma U}$$

$$= \frac{\frac{2}{s} - s}{\frac{2}{s} + 2 + s} = \frac{-s^2 + 2}{s^2 + 2s + 2} \tag{2-5-37}$$

$$Y_{22} = \frac{W_{1,1'}}{\Sigma U} = \frac{s^3 + 4s^2 + 2s + 4}{s^2 + 2s + 2} \tag{2-5-38}$$

(3) Voltage ratio transfer functions

$$\mu_{12} = \frac{\frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'})}{W_{1,1'}}$$

$$= \frac{s^2 - 2}{s^3 + 4s^2 + 2s + 4} \tag{2-5-39}$$

$$\mu_{21} = \frac{\frac{1}{2}(W_{1,2'} + W_{2,1'} - W_{1,2} - W_{1',2'})}{W_{2,2'}}$$

$$= \frac{s^2 - 2}{s^2 + 3s + 2} \tag{2-5-40}$$

(4) Current ratio transfer functions

$$\alpha_{12} = \frac{\frac{1}{2}(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'})}{W_{2,2'}}$$

$$= \frac{-s^2 + 2}{s^2 + 3s + 2} \qquad (2\text{-}5\text{-}41)$$

$$\alpha_{21} = \frac{\frac{1}{2}(W_{1,2} + W_{1',2'} - W_{1,2'} - W_{2,1'})}{W_{1,1'}}$$

$$= \frac{-s^2 + 2}{s^3 + 4s^2 + 2s + 4} \qquad (2\text{-}5\text{-}42)$$

3. Tree Finding Method

Most topological formulas for network analysis depend on the determination of all the trees of the network graph. The number of trees of a complex network may be greater than a million. Therefore, the computation time and memory space are the primary factors we have to consider when a computer is used for tree finding. Many tree-finding methods have been given by various authors [9] - [15]. But one or more disadvantages exists in each of these methods. They are listed in Table 3.

TABLE 3

Tree-finding method given by	Disadvantages
G. J. Minty	long computation time, since (2T-1) graphs are needed for T trees.
G. W. Zobrist and G. V. Lago	simplify the tree finding method for a complete graph only by one step.
W. Mayeda and S. Seshu	large memory space required, also tedious for hand calculation.
M. Piekassky	long computation time and large memory are required due to duplication.
Wai-Kai Chen	Same as above.
B. R. Myers and L. V. Auth	Same as above.
L. M. Maxwell and J. M. Cline	Same as above.

Two different tree-finding methods are given in this paper: Dichotomy method and T-triangle method. Both of these methods do not result in duplications, hence, these two methods achieve both a reduced computation time and a smaller memory requirements. However, the T-triangle method is much better for a complicated network.

3.1 Dichotomy Method

Algorithm

1. List the sets of edges incident to each of the vertices of the given graph and arrange them in any arbitrary sequences in a row (see examples) and call them "vertex sets."
2. Comparison Procedure
   (1) Take the first vertex as an "operating vertex." Compare its vertex set with one of any other vertex which has at least

one edge in common with that of the operating vertex. This other vertex is designated as 'comparing vertex'. List (store) the common edges found by this step as a "common set."

- (2) Create two new "vertex sets" in the following manner:
  - (a) Eliminate the common edges from both the operating vertex set and the comparing vertex set. Place them with the remaining ordered vertex sets to the right of the same row.
  - (b) Combine the operating vertex set and the comparing vertex set with their common edges eliminated to form a new "operating vertex" set, then place it together with the other original vertex sets below in the same column.

3. Repeat 2 successively for the new vertex sets through the same column until an empty set is resulted.
4. Return to the first non-zero vertex set, which was stored in the preceding rows, and repeat 2 and 3. Repeat this successively until an empty vertex set is found in the first row.
5. List all the trees by the following steps:
   - (1) Form an ordered group from all "common sets" in the first column. Form additional ordered groups by taking the common sets generated in each succeeding column and replacing them for the common sets in their respective rows in the preceding column until the common sets of the final column have been used.
   - (2) List all the trees from each group of common sets by taking one element at a time from each of the common sets.

Example 3.1

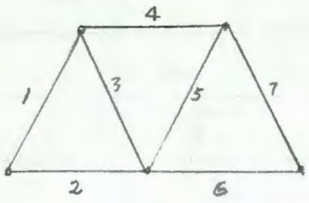


Fig. 3-1 The given graph with each edge designated by a number.

(I) List the vertex sets

[12] (134)(2356) (457) (67)
↑
Operating set.

(II) Comparison

```
[12](134)(2356)(457)(67) ------------------------------> ⓐ
 *   *
   1

[234](2356)(457)(67) -------> [4](56)(457)(67)
 **   **                       *      *
     23                           4

[456](457)(67) --> [6](7)(67)  [57](56)(67) --> [7](6)(67)
 **   **            *     *     *   *            *      *
     45               6           5                7

[67](67)           [7](7)      [67](67)         [6](6)
 **  **             *  *        **  **           *  *
   67                 7           67              6


ⓐ [2](34)(2356)(457)(67) ----------------------> [0] . . . . stop
   *      *
     2

  [356](34)(457)(67) ------> [56](4)(457)(67) -> [6](4)(47)(67)
   *    *                     *       *           *         *
     3                         5                  6

  [564](457)(67) --> [6](7)(67)  [647](4)(67)    [7](4)(47)
   * *  **             *     *      *   *          *      *
    45                 6            4              7

  [67](67)           [7](7)      [67](67)        [4](4)
   **  **             *  *        **  **          *  *
     67                 7           67             4
```

(III) Listing all the trees

Group of Common Sets	TREES
(1)(23)(45)(67)	1246 1256 1346 1356 1247 1257 1347 1357
(1)(23)(6)(7)	1267 1367
(1)(4)(5)(67)	1456 1457
(1)(4)(7)(6)	1467
(2)(3)(45)(67)	2346 2347 2356 2357
(2)(3)(6)(7)	2367
(2)(5)(4)(67)	2546 2547
(2)(6)(7)(4)	2674

Total number of trees = 21

Example 3.2

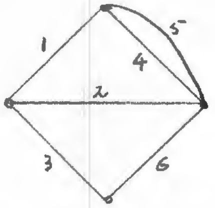


Fig. 3-2 The given graph with each edge designated by a number.

(I) List the vertex sets

[123] (145) (36) (2456)

(II) Comparison

$$[\underset{*}{1}23](\underset{*}{1}45)(36)(2456) \longrightarrow [2\underset{*}{3}](45)(\underset{*}{3}6)(2456) \rightarrow ⓐ$$

$$1 \qquad\qquad\qquad\qquad 3$$

$$[2\underset{*}{3}45](\underset{*}{3}6)(2456) \longrightarrow [\underset{*}{2}\underset{*}{4}\underset{*}{5}](6)(\underset{*}{2}\underset{*}{4}\underset{*}{5}6) \qquad [\underset{*}{2}\underset{*}{6}](45)(\underset{*}{2}45\underset{*}{6})$$

$$3 \qquad\qquad\qquad 245 \qquad\qquad\qquad 26$$

$$[\underset{*}{2}\underset{*}{4}\underset{*}{5}\underset{*}{6}](\underset{*}{2}\underset{*}{4}\underset{*}{5}\underset{*}{6}) \qquad [\underset{*}{6}](\underset{*}{6}) \qquad [\underset{*}{4}\underset{*}{5}](\underset{*}{4}\underset{*}{5})$$

$$2456 \qquad\qquad\qquad 6 \qquad\qquad\qquad 45$$

$$ⓐ\ [\underset{*}{2}](45)(6)(\underset{*}{2}456) \longrightarrow [0] \text{ stop}$$

$$2$$

$$[\underset{*}{4}\underset{*}{5}6](\underset{*}{4}\underset{*}{5})(6)$$

$$45$$

$$[\underset{*}{6}](\underset{*}{6})$$

$$6$$

(III) List all the trees

Group of Common Sets	Trees			
(1)(3)(2456)	123	134	135	136
(1)(245)(6)	126	146	156	
(3) (26)(45)	234	235	346	356
(2) (45) (6)	246	256		

Total number of trees = 13

Discussion

As illustrated by these two examples, we have the following conclusions to this tree-finding method.

(1) All the combinations which listed from each group of the common sets by procedure 5, are trees.

(2) No duplication of trees occurs.

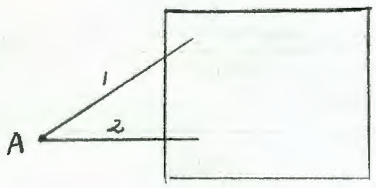
(3) All the trees of a given graph are listed.

The proof of these three statements are given below:

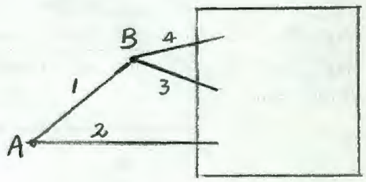
(a) The first common set are the edges common to the operating vertex and the comparing vertex, in other words, the corresponding edges in the first common set are incident to the two vertices of the graph. The elements in the second common set correspond to the edges, which are common to a new vertex and one of the prior two vertices (the vertices had been compared for obtaining the first common set). Similarly, the associated edges of the elements in every additional common set will connect a new additional vertex. Since there are V-1 (V is the number of vertices of a graph) common sets in each group, a combination, taking one element from each common set of one group, will connect V different vertices (all the vertices in a graph). Therefore, the edges collected by such combination is a tree.

(b) Since the new "vertex sets" stored in the $i+1^{th}$ column are generated by eliminating the common edges which have been found in the i^{th} column of the same row (see procedure 2-(2)-(a)), the elements in the common sets of the $i+1^{th}$ column will exclude some elements which are in one common set of the i^{th} column. Consequently, no trees list from different group of common sets will be the same. Thus, there is no tree duplications.

(c) We start our procedure by taking the first vertex of the original vertex sets to be the operating vertex; then we follow successive comparison procedure to find the common sets. This means that we take the edge associated to the elements in the operating vertex (say vertex A) as the reference edges and trace through them to the other vertices. The possible ways to reach other vertex (say vertex B) are through the edges which are common to both vertex A and vertex B. Other possible connections involve the edges corresponding to the elements, which are resulted from the ring sum operation of the vertex sets of A and B. This set of elements constitutes the new operating vertex by procedure 2-(2)-(b). The following Figures illustrate this point.



(a) Operating vertex A



(b) Common set $A\cap B = \{1\}$. New operating vertex set: $A\oplus B = \{2,3,4\}$

As we continue this process until one vertex set is empty, we get one group of common sets. Moreover, in each comparison step we also generate another new 'vertex set' with those elements of the common set (which have been found in the previous column) eliminated. By repeating this process repeatedly, we have all the possible groups of common sets. Thus, all the trees of the graph are found.

3.2 Tree Finding Computer Program I (by dichotomy method)

A digital computer program has been written in FORTRAN IV language for finding trees by the "dichotomy method." This program is included in Appendix I. Examples 3.1, 3.2 and 4-vertex to 8-vertex complete graphs are solved by the same program. General descriptions of the program are given below:

(1) Program Limitations.

- (a) Nonoriented linear graph.
- (b) Maximum number of vertices of a given graph is 15.
- (c) Maximum number of edges in a given graph is 60.

(2) Input Data.

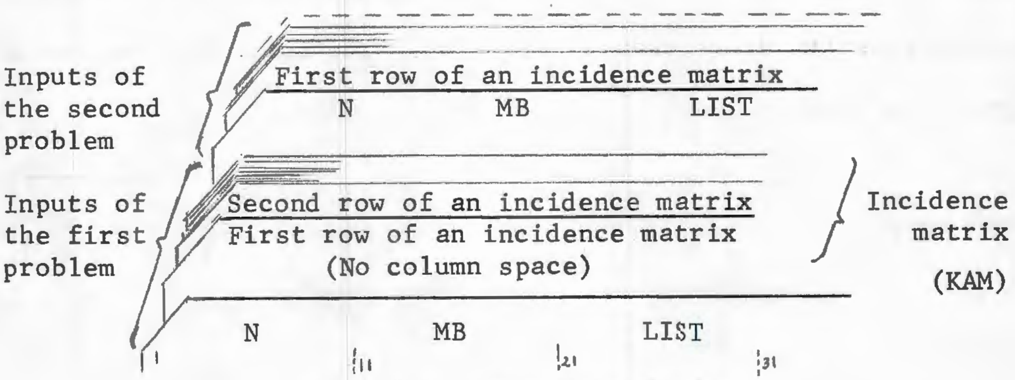

Inputs of the second problem
First row of an incidence matrix
N
MB
LIST
Inputs of the first problem
Second row of an incidence matrix
First row of an incidence matrix
(No column space)
Incidence matrix
(KAM)
N
MB
LIST
1
11
21
31

Fig. 3-3 Input Data Cards

N - Number of vertices of a given graph.
MB - Number of edges of a given graph.
LIST - Conditions for tree printing.

LIST = 1: print out all the trees that have been found.

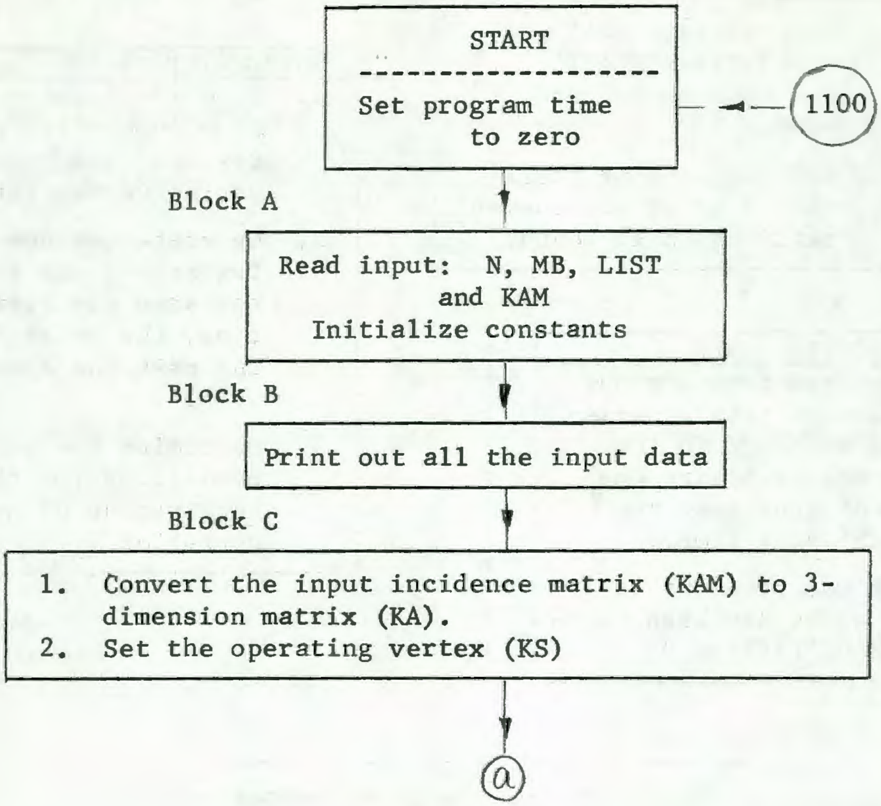
LIST = 0: do not print out the trees.

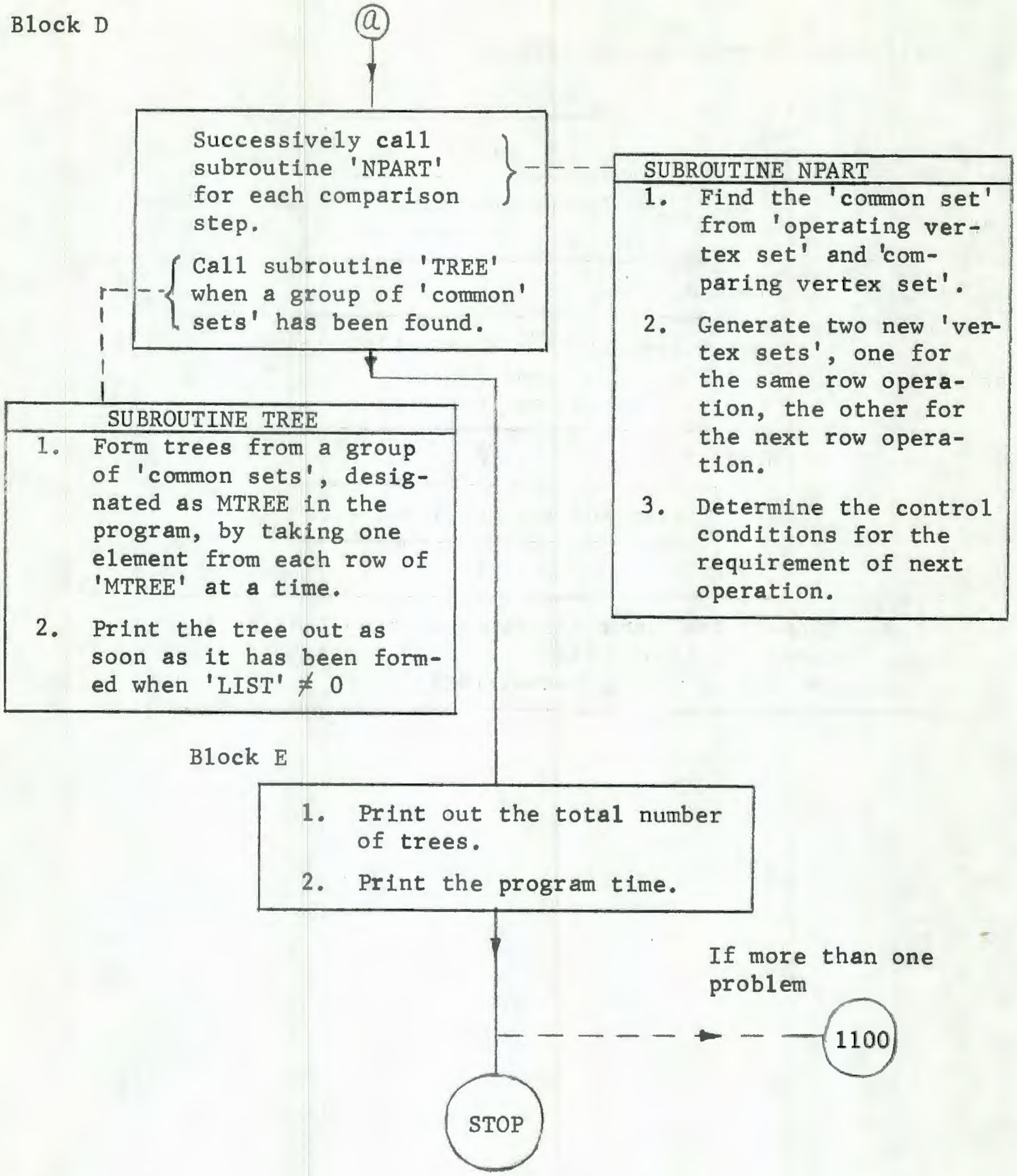
KAM -- the incidence matrix of a given graph.

(3) Outputs.

- (a) Trees, designated as 'IANS' in the program, can be printed out in the form of integer numbers which are labeled for each edge of the given graph.
- (b) Total number of trees, designated as 'KSUM' in the program.
- (c) Program time, designated as 'CL' in the program.

(4) Block diagram of the program.


START
Set program time to zero
1100
Block A
Read input: N, MB, LIST and KAM
Initialize constants
Block B
Print out all the input data
Block C
1. Convert the input incidence matrix (KAM) to 3-dimension matrix (KA).
2. Set the operating vertex (KS)
a



Block D

Successively call subroutine 'NPART' for each comparison step.

Call subroutine 'TREE' when a group of 'common' sets' has been found.

SUBROUTINE NPART

1. Find the 'common set' from 'operating vertex set' and 'comparing vertex set'.
2. Generate two new 'vertex sets', one for the same row operation, the other for the next row operation.
3. Determine the control conditions for the requirement of next operation.

SUBROUTINE TREE

1. Form trees from a group of 'common sets', designated as MTREE in the program, by taking one element from each row of 'MTREE' at a time.
2. Print the tree out as soon as it has been formed when 'LIST' ≠ 0

Block E

1. Print out the total number of trees.
2. Print the program time.

If more than one problem

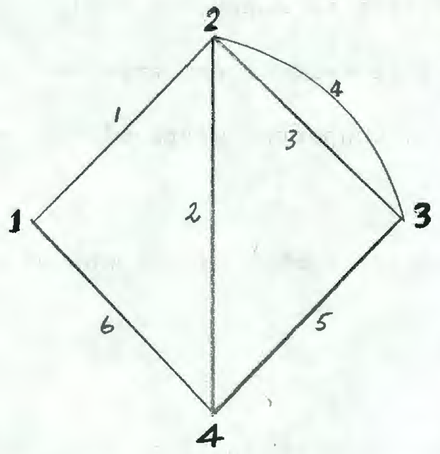
3.3 T-triangle Method.

This tree finding method is based on a T-triangle which is obtained from a given graph. We generate succeeding new T-triangles by the procedure, which will be stated in the algorithm. Then we list all the trees from each of the T-triangles. This method achieves both a reduced computation time and a smaller memory requirement for computer implementation. It is also suited to hand computation because no duplications occur and no new graphs are required.

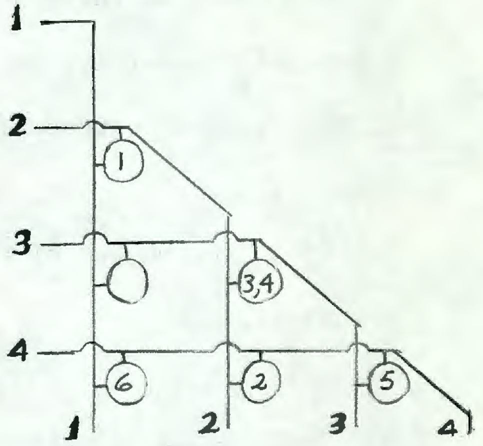
T-triangle. A T-triangle of a graph is a triangular array, of which the element t_{ij}, in the i^{th} row and j^{th} column of the T-triangle, is given by

$$t_{ij} \equiv \left\{ I_{i+1} \cap I_j \;\middle|\; \begin{matrix} i = 1, 2, \cdots, V-1 \\ j \le i \end{matrix} \right\}$$

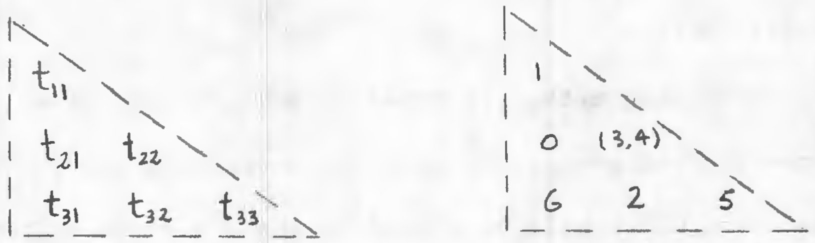
where I_K is the set of all the edges incident to vertex K and V is the number of vertices in the graph. Fig. 3-4 illustrates how the T-triangle of a graph is obtained.



(a)



(b)



(c) (d)

Fig. 3-4. Example illustrating T-triangles (a) a given graph G, (b) T-triangular diagram corresponding to G, (c) general form of a T-triangle corresponding to a 4-vertex graph, and (d) the T-triangle simplified from (b).

If we expand the vertices of a graph to lines, called vertex lines, and reduce all the edges between each pair of vertices to small circles and connect them to their corresponding vertex lines in a triangular form, we will obtain a diagram as shown in Fig. 3-4(b). Actually, Fig. 3-4(d) is a simplified form of Fig. 3-4(b). We can consider each circle, which includes all the parallel edges between a pair of vertices, as a single switch. When we say that the switch is non-empty, we imply that one or more elements (edges) exists in a circle. When we say that the switch is empty, we imply that no element exists in the circle. With this concept we can examine some properties of T-triangles corresponding to a connected graph with V vertices.

Property 3.1.

Both the number of rows and the number of columns of a T-triangle are (V-1).

Property 3.2.

The number of vertex lines of a T-triangle is V and the total number of entries (circles) in a T-triangle is V(V-1)/2.

Property 3.3

The switch corresponding to t_{ij} of a T-triangle has one end connected to vertex line (i+1) and the other end connected to a vertex line, which appears prior to the $(i+1)^{th}$ row in the triangular diagram (or the T-triangle).

Property 3.4

No two switches (sets of edges incident to two different pairs of vertices) of a T-triangle connect to the same pair of vertex lines.

Proof: From the definition of t_{ij} (an entry of a T-triangle), there is at least one subscript, either i or j, which is different from the corresponding subscript of any other entry.

Property 3.5

(a) All the switches corresponding to the entries of the first column of a T-triangle are connected to vertex 1.

(b) All the switches corresponding to the entries of the last row of a T-triangle are connected to vertex V.

(c) All the switches corresponding to the entries in the i^{th} row and the entries in the $(i+1)^{th}$ column of the T-triangle are connected to vertex (i+1).

Proof: The subscript j of each t_{ij} in the first column is 1; this implies that the corresponding switch has one end connected to vertex 1. The subscript i of each t_{ij} in the last row, the $(V-1)^{th}$ row, is V-1. Therefore, the corresponding switch has one end connected to vertex V. Each of the entries of the i^{th} row has a subscript i. This implies that the corresponding switch is connected to vertex (i+1). Similarly, a switch corresponding to an entry in the $(i+1)^{th}$ column is connected to vertex (i+1).

Property 3.6

A T-triangle represents a linear graph and any linear graph can be converted to a T-triangle.

The proof of this property follows from the definition of a T-triangle and the concepts of vertex lines and circles (switches) and property 3.5.

Definition 3.1: Simple Loop

A simple loop is a closed loop formed by two edges which are incident to the same pair of vertices of a graph.

Convention: The elements inside a switch shall not include the vertices to which they are incident in the graph. A switch is closed to vertex k implies that the elements inside the switch are connected to vertex k.

Theorem 3.1

When a switch is closed to its associated vertices,

(a) No loop will result if there is only one element (edge) in that switch.

(b) C_2^M simple loops will exist if there are M elements in that switch, where $M \geq 2$.

The proof is sufficiently obvious to be omitted.

Theorem 3.2

No loop can result if only two switches in a T-triangle are closed to their associated vertices except those simple loops formed within the individual switches.

Proof: By property 3.4 we know that two switches will connect at least three different vertex lines, therefore, the corresponding subgraph is a subtree.

Theorem 3.3

If only one switch of each row preceding row i has been closed to its associated vertices, then

(a) No loop occurs except the simple loops when any one switch of the i^{th} row is closed to its associated vertices.

(b) Loops will result when two or more switches of the i^{th} row are closed to their associated vertices.

Proof: In the first row there are two vertex lines and the inclusion of each succeeding row increased the number of vertex lines by one. Thus, there are (i+1) vertex lines when the i^{th} row is included. From properties 3.3 and 3.4, the (i+1) vertex lines are only connected by i switches. Therefore, no loop exists, except possible for simple loops. If two or more switches are closed in the i^{th} row, then (i+1) vertex lines are connected by at least (i+1) switches, hence one or more loops exist.

Theorem 3.4

The collection of the elements, taking one element (edge) from each row of a T-triangle at a time, is a complete set of branches of a tree.

Proof: An element in row i represents an edge which connects vertex i+1 and one of the vertices preceding i+1. By taking one element from each row, each set contains V-1 edges, which connect V vertices. Thus, the theorem is true.

For example, the trees listed from the T-triangle in Fig. 3-4(d) by this theorem are shown in Table 4.

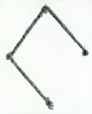
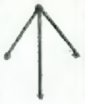
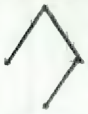
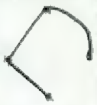
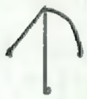
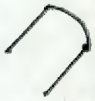
Combinations	136	132	135	146	142	145
Subgraphs						

Table 4 Trees list from the T-triangle in Fig. 3-4(d)

Theorem 3.5

If all the entries of row i (for i ≥ 2) and those directly below row i down to the bottom row are empty then the graph corresponding to the T-triangle is disconnected.

Proof: By property 3.3 the edges which can be connected to one of the vertices preceding vertex i + 1 are those elements in the ith row or directly below them. Since all of these elements are empty, the graph is disconnected.

An example for this theorem is illustrated in Fig. 3-5.

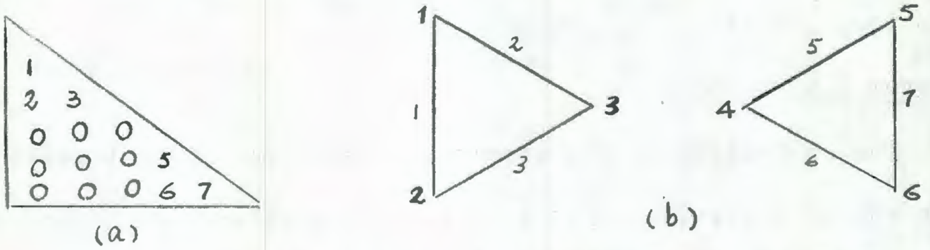


Fig. 3-5 Illustration for theorem 3.5

Theorem 3.6

If a T-triangle, T_1, is generated from an original T-triangle, T_o, according to steps (a) through (d),

(a) Move all the entries of the ith row of T_o directly down to the bottom row, and

(b) Shift all the entries, which are located directly below the ith row, upward one location, and

(c) Shift all the entries, which are located to the right of column (i+1), upward one location and toward the left by one location, and

(d) The entries in column (i+1) are translated, successively, downward to the bottom row and to the extreme right of that row. (Note that all the entries prior to the i^{th} row are unchanged.)

The graph G_1 corresponding to T_o are 2-isomorphic.

Proof: Since the procedure for the generation of the new T-triangle results in the rearrangement of vertex representation in the new T-triangle, graph G_1 and G_o are identical with the exception of different vertex sequences. Thus, they are 2-isomorphic.

The illustrations of this theorem are shown in Fig. 3-6.

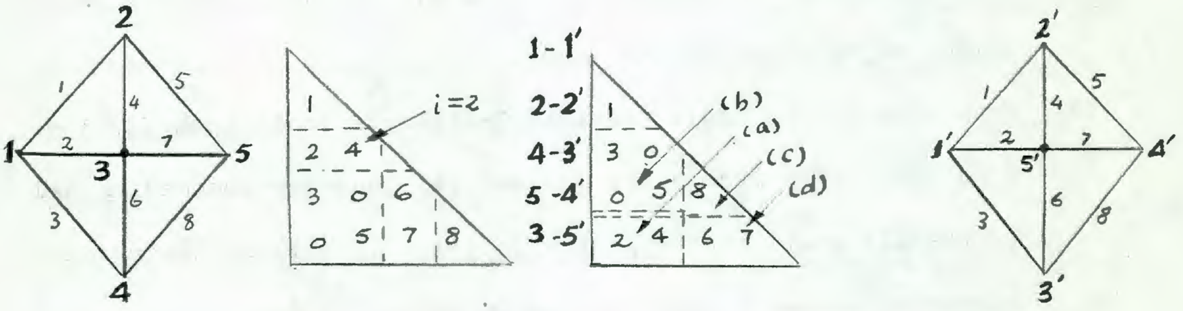


Fig. 3-6 Illustration for Theorem 3.6

Algorithm

The procedure for tree finding by the T-triangle method is given below.

(1) Obtain an original T-triangle, T_o, from the given graph.

(2) Take any column (call it column j) of T_o as an "operating column," provided that

(a) the entries of that column are not all zero, or

(b) the entries to the left of column j and below row (j-1) are not all zero.

(3) Obtain a new T-triangle, T_j, from T_o by

(a) removing the elements of the $(j-1)^{th}$ row of T_o, and

(b) shifting all the entries, which are located at the left of column j and below row (j-1), upward one location, and

(c) shifting all the entries at the right hand side of column j upward one location and toward the left by one location. Simultaneously the entries of the column j are translated, successively, downward to the bottom row and to the extreme right end of the T-triangle.

(4) Repeat (2) and (3) for every j = 2, 3, ... V-1, where V is the number of vertices of the graph.

(5) From each of the newly obtained T-triangle (call them T_j, j = 2, 3 ... , V-1) from steps (3) and (4) generate succeeding new T-triangle (call them T_{jk}, k = j, j+1, ... V-1) by taking an operating column k each time as in steps (2) and (3).

(6) Generate all the T-triangles successively by repeating (5) on each newly generated T-triangle.

(7) List all the trees from each T-triangle by obtaining all possible combinations, taking one element at a time from each row of a T-triangle.

The illustrations for each step stated in the above procedure are given in Fig. 3-7 through Fig. 3-10.

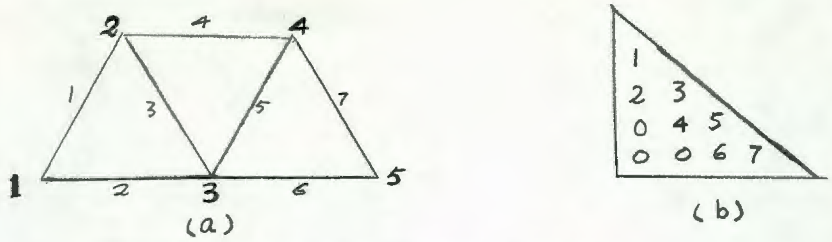


Fig. 3-7 Illustration for Step (1)

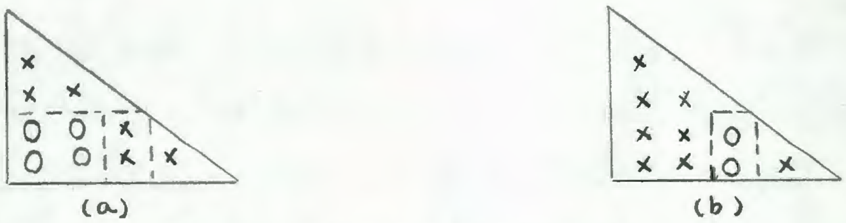


Fig. 3-8 Illustration for Step (2)

In either (a) or (b) column 3 cannot be the operating column.

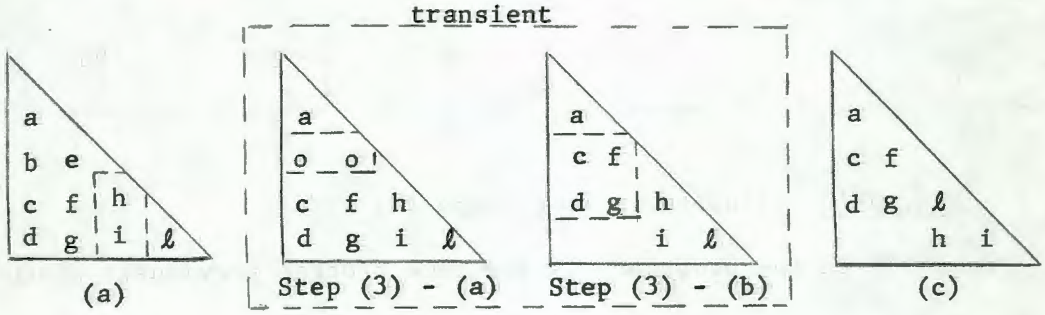


Fig. 3-9 Illustration for Step (3)

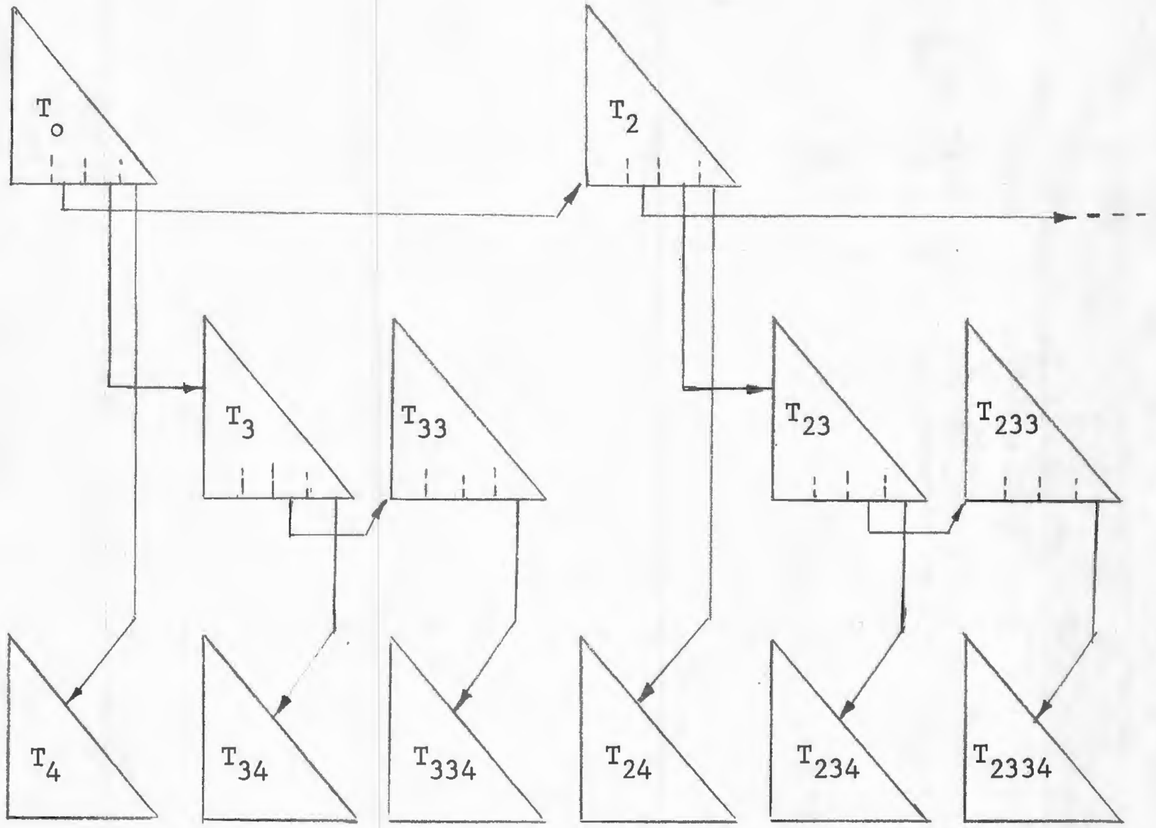


Fig. 3-10 Illustration for Steps (4) - (6)

Step (7) in the procedure is the same process previously discussed in theorem 3.4.

Occasionally we may encounter a special case that all the entrees in one row of T_o or in any new T-triangle are empty. (This case occurs when a given graph has a large number of vertical but with many pairs of which are unconnected.) If such a case occurs, in the above procedure, we must move this empty row immediately down to the bottom row as provided by theorem 3.6. Continue this procedure until no row is empty, then continue the other steps. This will be demonstrated in example 3.5.

Example 3.3

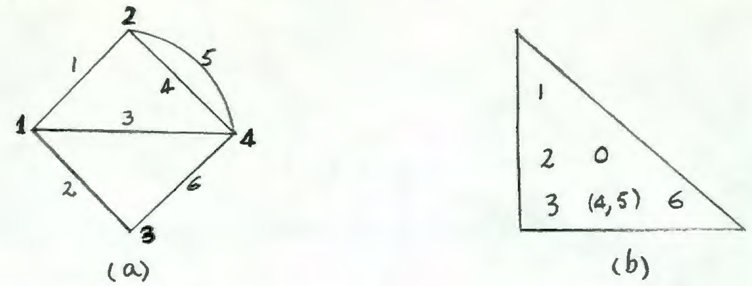
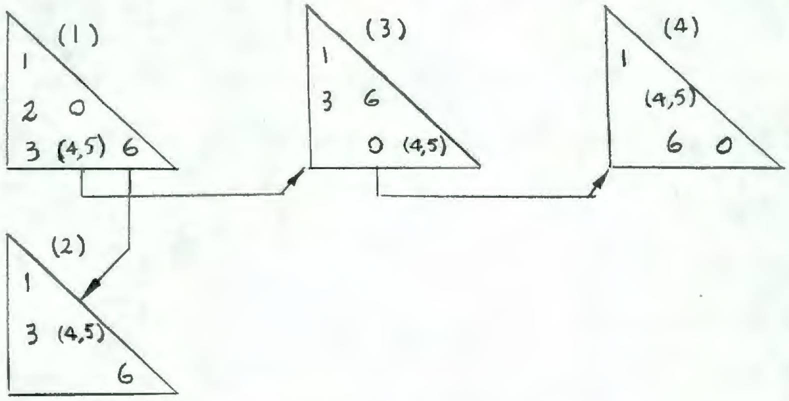


Fig. 3-11 (a) A given graph, (b) T_o, the T-triangle corresponding to the graph in (a).

(I) Generate all T-triangles



(II) List the trees of each T-triangle

(1) 123, 124, 125, 126

(2) 136, 146, 156

(3) 134, 135, 164, 165

(4) 346, 356

Total number of trees = 13

Example 3.4

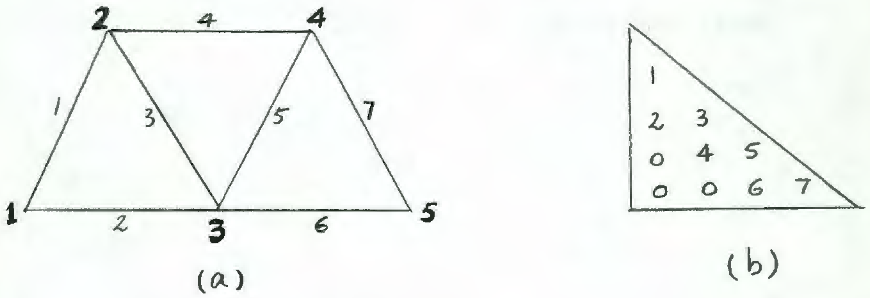
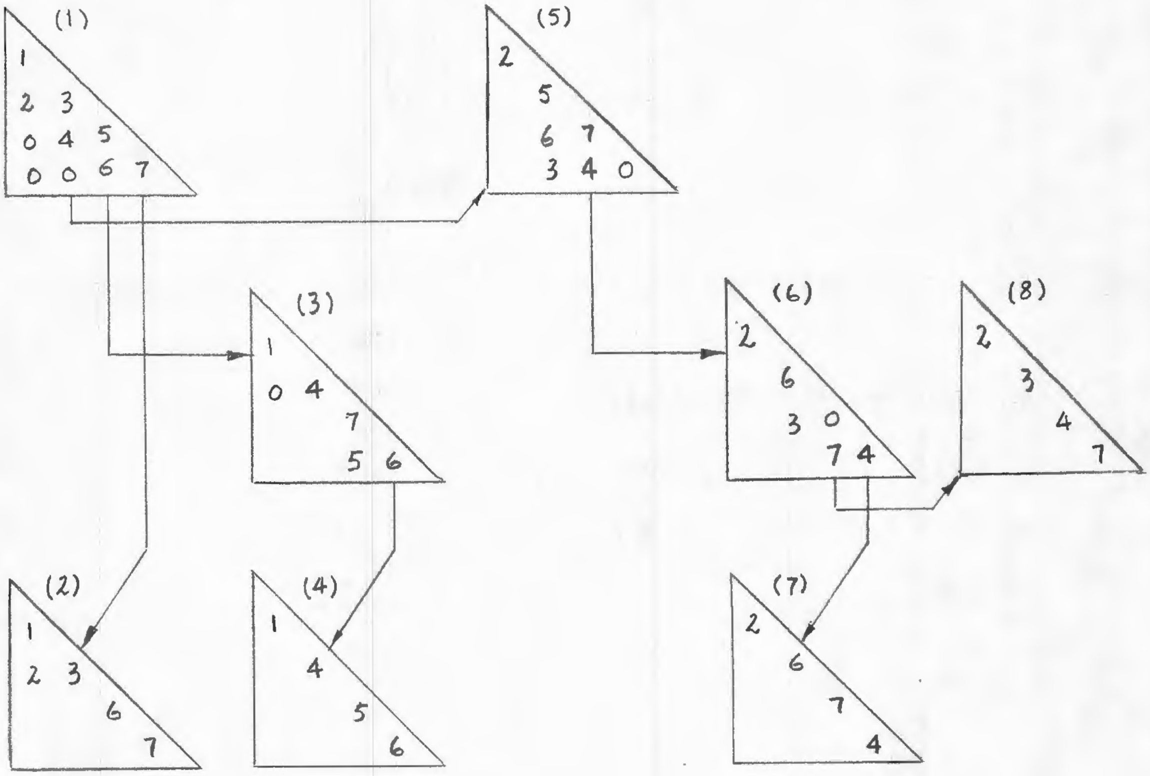


Fig. 3-12 (a) A given graph, (b) T_o

(I) Generate all T-triangles



(II) List the trees of each T-triangle

(1) {1246 1346
1247 1347
1256 1356
1257 1357

(2) {1267
1367

(3) {1475
1476

(4) {1456

(5) {2563
2564
2573
2574

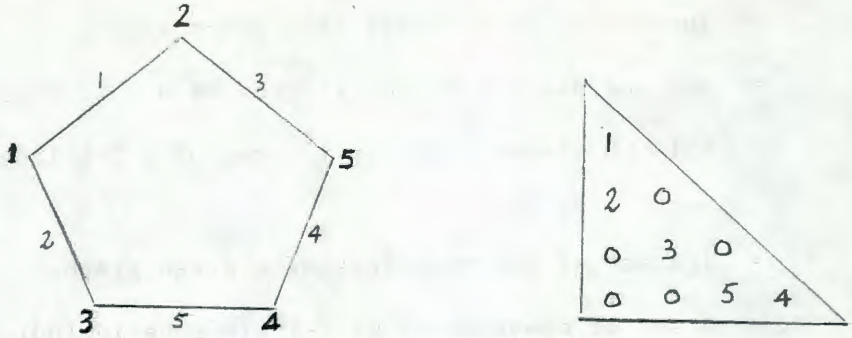
(6) {2637
2634

(7) {2674

(8) {2347

Total number of trees = 21

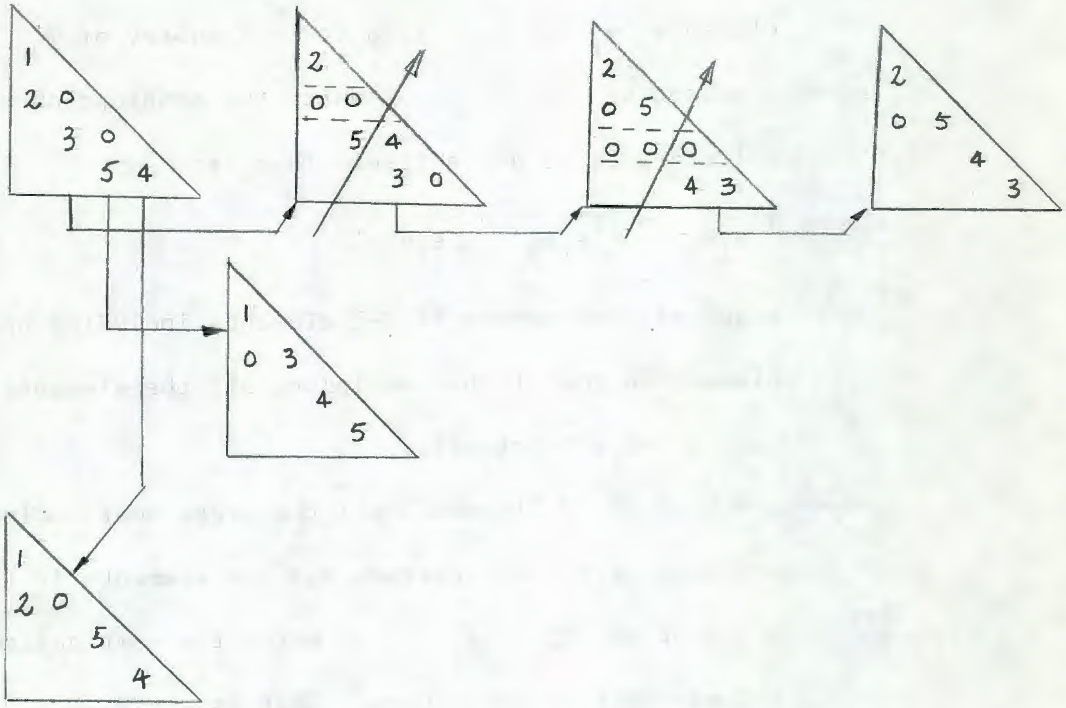
Example 3.5



(a) Graph G (b) T_0 corresponding to G

Fig. 3-13

(I) Generate all T-triangles



(II) List the trees of each T-triangle

(1) $\begin{cases} 1235 \\ 1234 \end{cases}$ (2) 1254 (3) 1345 (4) 2543

Total number of trees = 5

Proof of the validity of the tree finding method by T-triangle

(1) Definitions

V = The number of vertices of a given graph.

e_i = Any one element in the i^{th} row of a T-triangle.

E_i = All the elements in the i^{th} row of a T-triangle, i.e., $E_i \equiv \{ e_i \}$.

T = The set of all the trees of a given graph.

$T'_{e_i e_j}$ = A set of combination of V-1 elements including one element in row i and one element in row j of a T-triangle.

$T_{e_i e_j}$ = A sub set of T in which all the trees include elements e_i and e_j. Also it is a subset of $T'_{e_i e_j}$.

$\overline{T}_{e_i e_j}$ = A subset of $T'_{e_i e_j}$, of which the combination of V-1 elements is not a tree. That is:

$$T'_{e_i e_j} = T_{e_i e_j} \cup \overline{T}_{e_i e_j} .$$

$T'_{e_i - E_j}$ = A set of combination of V-1 elements including one element in row i but excluding all the elements in row j of a T-triangle.

$T_{e_i - E_j}$ = A subset of T in which all the trees must include the element e_i but exclude all the elements in E_j.

$\overline{T}_{e_i - E_j}$ = A subset of $T'_{e_i} - E_j$, of which the combinations of V-1 elements is not a tree. That is:

$$T'_{e_i - E_j} = T_{e_i - E_j} \cup \overline{T}_{e_i - E_j} .$$

(E_i) = All the possible combinations forms by taking k elements at a time from the set E_i. Where k = 1, 2, ... up to the total number of elements in E_i.

(2) Proof:

All the trees of a graph can be sorted out into separate groups according to the elements from which they are formed. If we sort T into two subsets such that all the trees containing element "b" form one subset and all the trees which do not contain element "b" form another subset, then T is the sum of the trees of these two subsets. We shall sort the trees of the graph by recognizing the elements included in each row of a T-triangle. For simplicity, we assume that every entry, t_{ij}, of a T-triangle has no more than one element in our proof. Notice that no generality is lost by this assumption, because when listing the trees from a T-triangle, we can consider the possibility of parallel edges in each entry of a T-triangle. We observe that

$$T = T_{(E_1)} + T_{-E_1} \tag{3-3-1}$$

But $\quad E_1 = \{ t_{11} \}$

Thus $\quad (E_1) = e_1 \quad$ (3-3-2)

substituting (3-3-2) into (3-3-1) we have

$$T = T_{e_1} + T_{-E_1} \tag{3-3-3}$$

Next, T_{e_1} can be expressed as

$$T_{e_1} = T_{e_1(E_2)} + T_{e_1 - E_2} \tag{3-3-4}$$

Notice $\quad (E_2) = \{ t_{21}, t_{22}, t_{21}t_{22} \}$, so that

$$T'_{e_1(E)_2} = T'_{e_1 t_{21} - t_{22}} + T'_{e_1 t_{22} - t_{21}} + T'_{e_1 t_{21} t_{22}}$$

$$= \left(T_{e_1 t_{21} - t_{22}} + \bar{T}_{e_1 t_{21} - t_{22}} \right) + \left(T_{e_1 t_{22} - t_{21}} + \bar{T}'_{e_1 t_{21} - t_{22}} \right) + T'_{e_1 t_{21} t_{22}} \tag{3-3-5}$$

The last term in (3-3-5) contains no tree at all according to theorem 3.3, hence $T'_{e_1 t_{21} t_{22}} = \bar{\bar{T}}_{e_1 t_{21} t_{22}}$, and we can rewrite (3-3-5) as:

$$T'_{e_1 (E_2)} = T_{e_1 t_{21} - t_{22}} + T_{e_1 t_{22} - t_{21}} + \bar{\bar{T}}_{e_1 (E_2)} \tag{3-3-6}$$

By definition we can write

$$T_{e_1 e_2} = T_{e_1 t_{21} - t_{22}} + T_{e_1 t_{22} - t_{21}} \tag{3-3-7}$$

Substituting (3-3-7) into (3-3-6), we have

$$T'_{e_1 (E_2)} = T_{e_1 e_2} + \bar{\bar{T}}_{e_1 (E_2)} \tag{3-3-8}$$

But
$$T'_{e_1 (E_2)} = T_{e_1 (E_2)} + \bar{\bar{T}}_{e_1 (E_2)} \tag{3-3-9}$$

Comparing (3-3-8) with (3-3-9), we obtain

$$T_{e_1 (E_2)} = T_{e_1 e_2} \tag{3-3-10}$$

Substituting (3-3-10) into (3-3-4), we have

$$T_{e_1} = T_{e_1 e_2} + T_{e_1 - E_2} \tag{3-3-11}$$

Continue sorting $T_{e_1 e_2}$, we get

$$T_{e_1 e_2} = T_{e_1 e_2 (E_3)} + T_{e_1 e_2 - E_3} \tag{3-3-12}$$

By similar manipulations we can drive

$$T_{e_1 e_2 (E_3)} = T_{e_1 e_2 e_3} \tag{3-3-13}$$

Substituting (3-3-13) into (3-3-12), we have

$$T_{e_1 e_2} = T_{e_1 e_2 e_3} + T_{e_1 e_2 - E_3} \tag{3-3-14}$$

Substituting (3-3-14) into (3-3-11) and then into (3-3-3), we have

$$T = T_{e_1 e_2 e_3} + T_{e_1 e_2 - E_3} + T_{e_1 - E_2} + T_{-E_1} \tag{3-3-15}$$

If we continue to sort the first term on the right hand side of equation (3-3-15) by the same manipulations until E_{v-1}, the elements of the last row of the T-triangle, have been considered, we will have

$$\begin{aligned} T = {} & T_{e_1 e_2 \cdots e_{v-1}} + T_{e_1 e_2 \cdots e_{v-2} - E_{v-1}} + T_{e_1 e_2 \cdots e_{v-3} - E_{v-2}} + T_{e_1 e_2 \cdot\cdot e_{v-4} - E_{v-3}} \\ & + \cdots\cdots + T_{e_1 e_2 - E_3} + T_{e_1 - E_2} + T_{-E_1} \end{aligned} \tag{3-3-16}$$

Now, let us examine each term on the right hand side of (3-3-16). We see that the first term gives the trees that we can list from T_o by step (7) as stated in the algorithm. The second term is an empty set (no complete tree exists) because of removal of all the elements of the last row (corresponding to the removal of a node from the graph; see property 3.5). The third term gives the trees that we can list from T_{V-1}. (T_{V-1} is defined in step (5) of the algorithm.) The fourth term gives the trees that we can obtain from the T-triangle, T_{V-2}, which is obtained from T_o by removing the elements in row (V-3) and keeping the elements in rows 1, 2, ... V-4 unchanged (in order that we can have one element from each of these rows). Now, with all the entries of row (V-3) removed, we take column j = V-2 as the operating column and following step (3) of the algorithm to generate the new T-triangle, T_{V-2}. The trees in the set $\mathcal{T}_{e_1 e_2 \cdots e_{V-4} - E_{V-3}}$ are all the trees given by T_{V-2}. Similarly, we can sort the set $\mathcal{T}_{e_1 e_2 \cdots\cdots e_{V-4} - E_{V-3}}$ associated with T_{V-2} and the results will be the trees that we can list from $\mathcal{T}_{V-2}$, $\mathcal{T}_{V-2,V-1}$, $\mathcal{T}_{V-2,V-2}$ and $\mathcal{T}_{V-2,V-2,V-1}$ by step (7) of the algorithm.

By repeating the same manipulation for the other terms in equation (3-3-15), we will have

$$T = \left\{ \begin{array}{l} \text{The trees given by the original T-triangle, } T_o \text{, and} \\ \text{all the new T-triangles, which are generated by steps} \\ \text{(2) - (6), with tree listing method given in step (7)} \\ \text{in the algorithm.} \end{array} \right\}$$

This completes the proof.

3.4 Tree Finding Computer Program II (by T-triangle method)

A digital computer program has been written in FORTRAN IV language for finding trees by T-triangle method. This program is included in Appendix II. Example 3.3, 3.4, and the 4-vertex to 8-vertex complete graphs are solved by this program. The general descriptions of this program are given below.

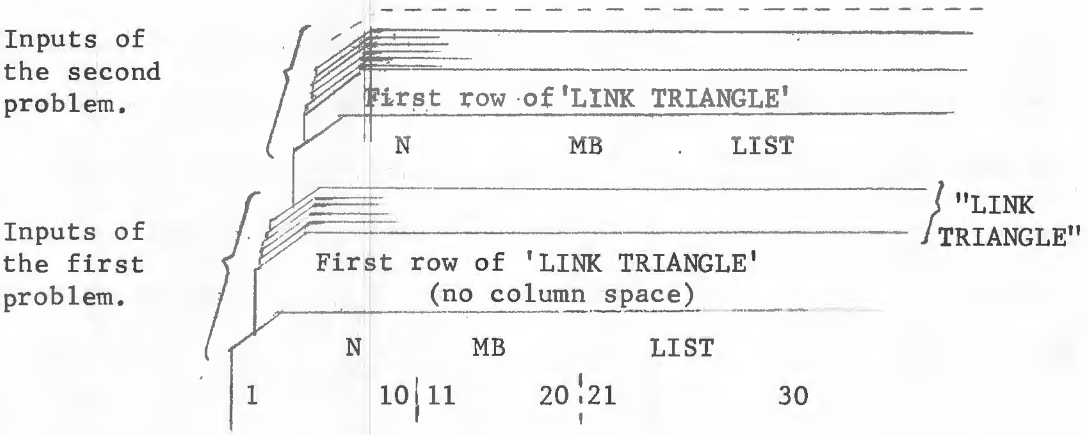
(1) Program limitations.

(a) non-oriented linear graph

(b) maximum number of vertices of a given graph is 15

(c) maximum number of edges in a given graph is 300.

(2) Input data.



Inputs of the second problem.
First row of 'LINK TRIANGLE'
N
MB
LIST
"LINK TRIANGLE"
Inputs of the first problem.
First row of 'LINK TRIANGLE' (no column space)
N
MB
LIST
1
10
11
20
21
30

Fig. 3-14 Input data cards

N - Number of vertices of a given graph.

MB - Number of edges of a given graph.

LIST - Conditions for tree printing.

LIST = 1: print out all the trees that have been found.

LIST = 0: do not print out the trees.

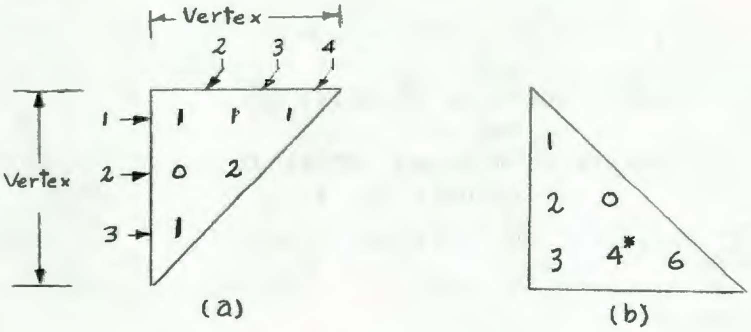
The "Link triangle" of a graph is a triangular array, the element of which, ℓ_{ij} , in the i^{th} row and j^{th} column of the "Link triangle" is given by

$\ell_{ij} \equiv$ the number of edges between vertices i and $(i+j)$.

Where $i = 1, 2, \ldots V-1$, and V is the number of vertices of the graph.

$j + i \leq V$

an example of the "Link triangle" corresponding to Fig. 3-11(b) is shown in Fig. 3-15(a).



(a)

(b)

Fig. 3-15 (a) 'Link Triangle' corresponding to Fig. 3-11(b)

(b) T-triangle generated from (a) by the computer program.

From the input of "LINK" the program itself will generate the original T-triangle, T_o, and label the elements in T_o by integers in the sequence order as shown in Fig. 3-15(b). Notice that there are two edges between vertices 2 and 4 which are indicated by ℓ_{22}, (i.e., edges #4 and #5) but edge #5 disappears in T_o.

However, the computer will recover those parallel edges which are temporarily omitted from the representation when the trees are listed from the T-triangle later.

(3) Output.

(a) Trees, designated as 'ITREE' in the program, can be printed out in the form of integers which are used for labeling the edges of the given graph.

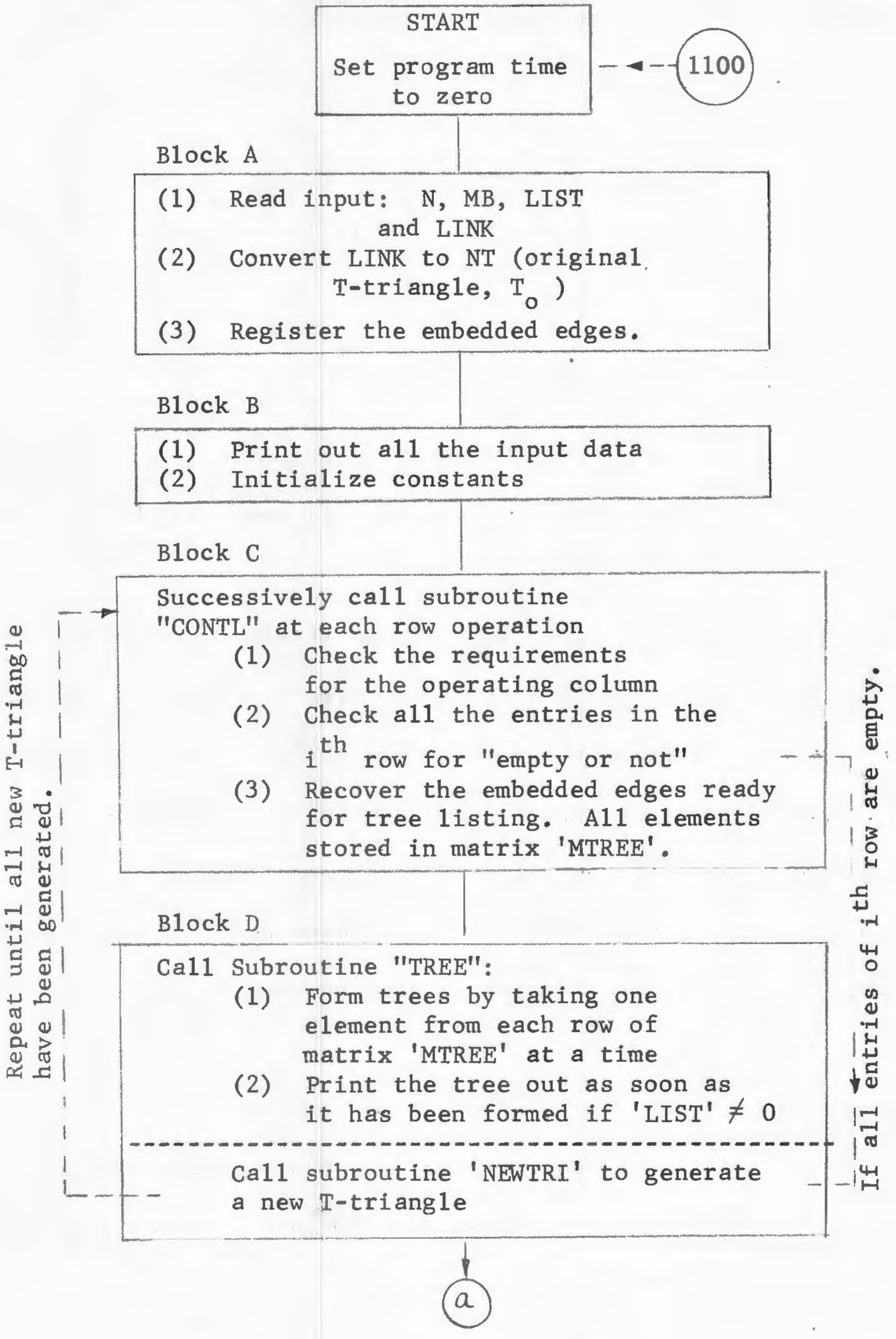
---

* edge #5 is embedded in #4.

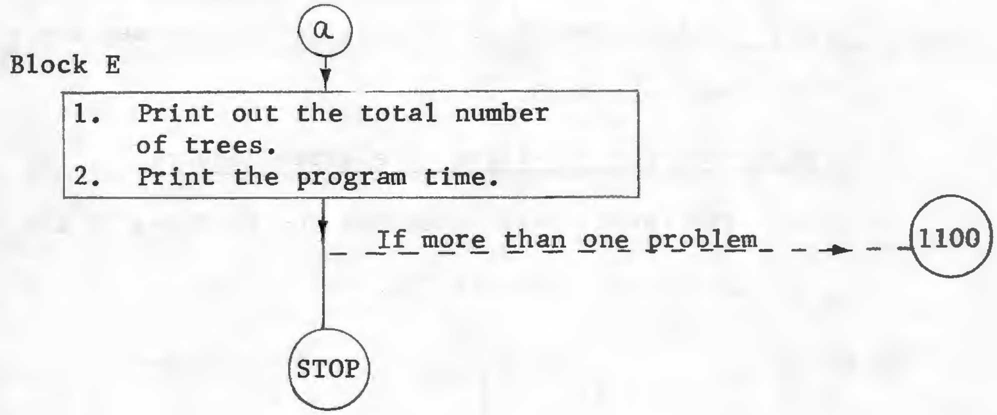
(b) Total number of trees.

(c) Program time.

(4) Block diagram of the program.



START
Set program time to zero
1100
Block A
(1) Read input: N, MB, LIST and LINK
(2) Convert LINK to NT (original T-triangle, T_o)
(3) Register the embedded edges.
Block B
(1) Print out all the input data
(2) Initialize constants
Block C
Successively call subroutine "CONTL" at each row operation
(1) Check the requirements for the operating column
(2) Check all the entries in the i^{th} row for "empty or not"
(3) Recover the embedded edges ready for tree listing. All elements stored in matrix 'MTREE'.
Block D
Call Subroutine "TREE":
(1) Form trees by taking one element from each row of matrix 'MTREE' at a time
(2) Print the tree out as soon as it has been formed if 'LIST' $\neq$ 0
Call subroutine 'NEWTRI' to generate a new T-triangle
Repeat until all new T-triangle have been generated.
If all entries of i^{th} row are empty.
a


a
Block E
1. Print out the total number of trees.
2. Print the program time.
If more than one problem
1100
STOP

4. Computer Program for the Determination of Network Functions.

A computer program for the determination of network functions, i.e., driving point admittance, driving point impedance, open circuit impedance, short circuit admittance, voltage ratio transfer function and current transfer function, has been written in FORTRAN IV languate and it is included as Appendix III. The T-triangle method of tree finding, which has been already described in section 3.3, is inserted in this program as a principal subprogram. The dichotomy method of tree finding is easier to handle a network which has parallel edges between any pair of vertices; but, it requires more memory space than the T-triangle method does. Also, the time of computation for the former will be a little longer if the network is more complex. The same networks given in examples 2.1 and 2.3 were solved using this program.

4.1 Program limitation

(1) Passive network without mutual inductance

(2) The number of vertices of a given network are limited from 3 to 15.

(3) Maximum number of edges of a given network is 300.

(4) Maximum number of parallel edges between any pair of vertices is 5.

4.2 Rules for the labelling of a given network

(1) The labels designated for the vertices of the terminal ports are shown in Fig. 4.1.

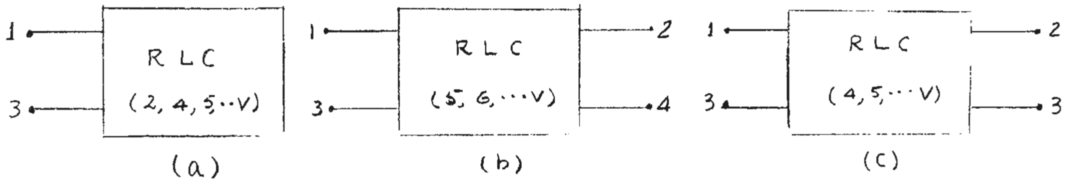


Fig. 4-1 (a) 1-P (b) 2-P (c) 2-p, when vertices 3 and 4 are shorted together.

The remaining vertices are labeled with the numbers shown in parenthesis in Fig. 4-1, where V is the number of vertices of the given network.

(2) The labels of edges are designated by the integers from 1 through E, the total number of edges. The parallel edges between a pair of vertices must be numbered in sequence order.

4.3 Input data.

The data card configurations required for the running of the program are described below.

Data Card Sequence No.	Columns	Description and Specifications
1	1 2 3	Blank, Dash and Plus Sign; used for printing the dividing line and signs of the output.

Sequence No.	Columns	Description and Specifications
	4-72	Disregarded.
2	1-60	Titles.
	61-72	Disregarded.
3	1-50	Titles.
	51-72	Disregard.
4	1-60	Titles.
	61-72	Disregard.
5	1-5	Number of vertices; Max: 15
	6-10	Number of edges; Max: 300
	11-15	Ground conditions; 1 if Fig. 4.1(c) is met. 2 if Fig. 4.1(b) is met. (No constrain for 1-P)
	16-20	Port conditions; 1: 1-P 2: 2-P
	21-72	Disregarded.
6	1-10	Same integer as labeled for the edge in a given network.
	11-13	Vertices connected by the
	14-16	edge as indicated in column 1-10. The vertex number designated in column 14-16 must be the larger one.
	17-21	Same integer as designated for the corresponding entry of the original T-triangle.
	22-26	Number of parallel edges between the vertices as given in columns 11-16.
	27-31	Sequence number of the edge between its parallel edges.
	32-41	Magnitude of the admittance of the edge; in decimal point form (F10.4).

Sequence No.	Columns	Description and Specifications
	42-46	Degree in S-domain of the admittance of the edge.
	47-72	Disregarded.

The configurations of the remaining data cards of a problem are the same as card No. 6 and each card specifies one different edge. The input data cards for examples 2.1 and 2.3 are shown in Table 5.

1

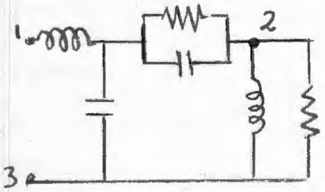
Title

(Cards 1-5 are the same for any given network.)

2

Input data for

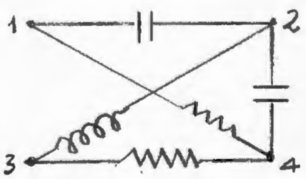
Example 2.1



3

Input data for

Example 2.3



```
  1         11        21        31        41        51
   +-
    DRYG Y=   Z11    = Z12=Z21=   Z22    =   Y11   =  Y12=Y21=
1   Y22    =  V2/V1  =  V1/V2  =  I2/I1  =   I1/I2 =
    V(Y)     W1,1'(Y)  W2,2'(Y)  W1,2(Y)    W1',2'(Y) W1,2'(Y)
   W2,1'(Y)  U(Y)
      4    6    1    1
           1  1  4    1    1    1     1.       -1
           2  3  4    2    1    1     2.        1
2          3  2  4    3    2    1     1.        0
           4  2  4    3    2    2     3.        1
           5  2  3    4    2    1     1.       -1
           6  2  3    4    2    2     2.        0
      4    5    2    2
           1  1  2    1    1    1     1.        1
3          2  2  3    2    1    1     2.       -1
           3  1  4    3    1    1     1.        0
           4  2  4    4    1    1     1.        1
           5  3  4    5    1    1     1.        0
```

Table 5 Input data for example 2.1 and 2.3

4.4 Outputs.

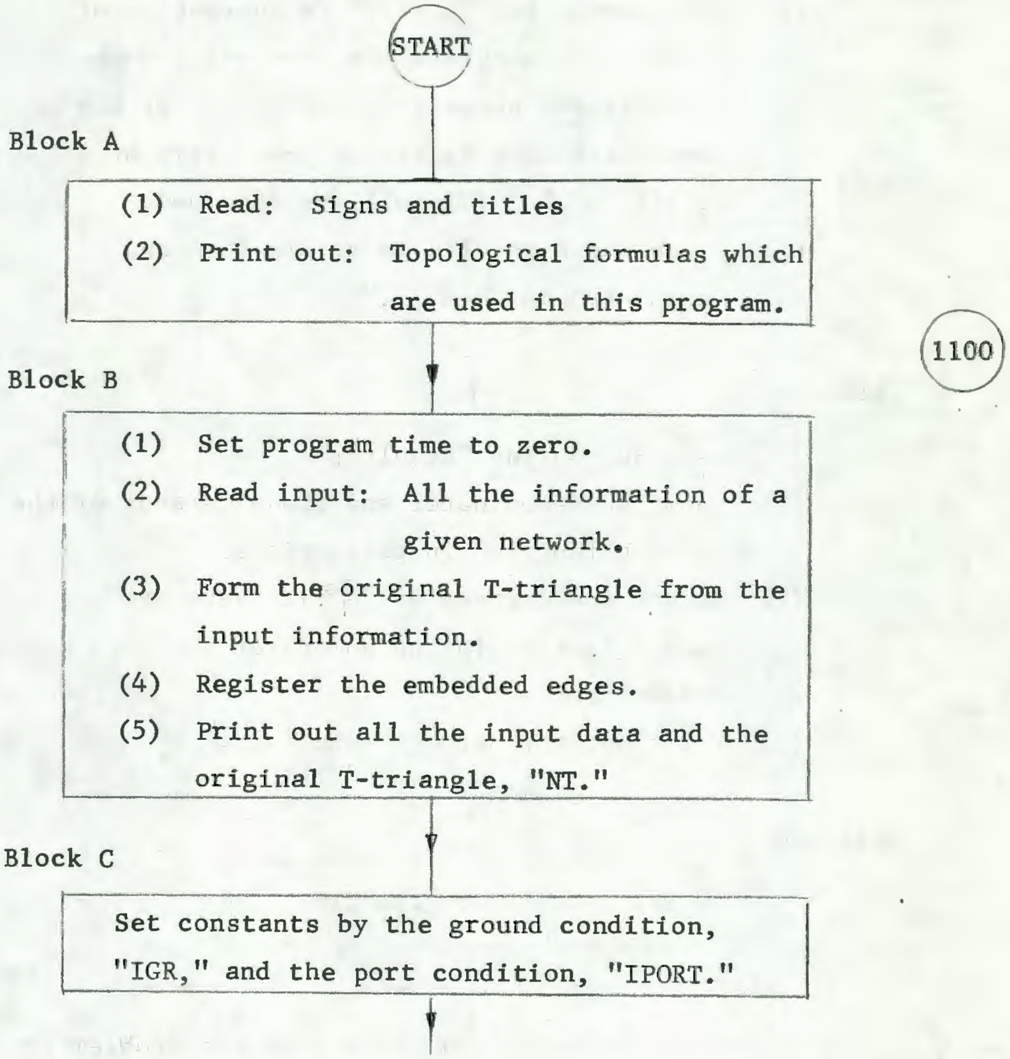
(1) Trees and tree admittance products of the given network and modified network will be printed out for the purpose of checking.

(2) All the network functions of 1-P or 2-P will be printed out in the regular form as shown in the following example:

$$\text{DRYG Y} = \frac{2.0S{*}{*}0 + 6.0S{*}{*}1 + 4.0S{*}{*}2 + 1.0S{*}{*}3}{4.0S{*}{*}0 + 2.0S{*}{*}1 + 4.6S{*}{*}2 + 1.0S{*}{*}3}$$

The integer which follows "**" is the power of "S."

4.5 Block diagram of the program



Block A

(1) Read: Signs and titles

(2) Print out: Topological formulas which are used in this program.

Block B

(1) Set program time to zero.

(2) Read input: All the information of a given network.

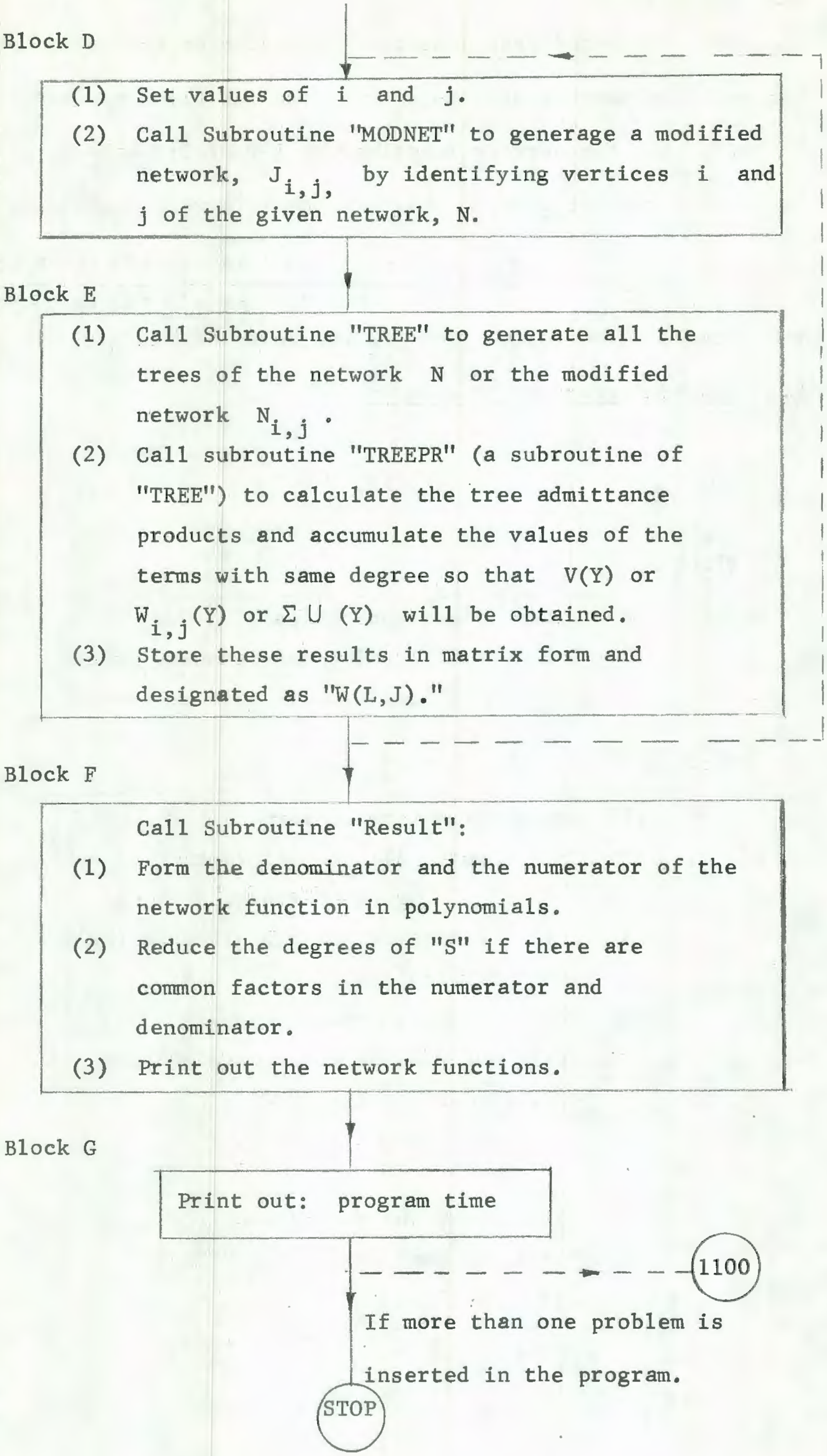
(3) Form the original T-triangle from the input information.

(4) Register the embedded edges.

(5) Print out all the input data and the original T-triangle, "NT."

Block C

Set constants by the ground condition, "IGR," and the port condition, "IPORT."



Block D

(1) Set values of i and j.

(2) Call Subroutine "MODNET" to generage a modified network, $J_{i,j}$, by identifying vertices i and j of the given network, N.

Block E

(1) Call Subroutine "TREE" to generate all the trees of the network N or the modified network $N_{i,j}$.

(2) Call subroutine "TREEPR" (a subroutine of "TREE") to calculate the tree admittance products and accumulate the values of the terms with same degree so that V(Y) or $W_{i,j}(Y)$ or $\Sigma U(Y)$ will be obtained.

(3) Store these results in matrix form and designated as "W(L,J)."

Block F

Call Subroutine "Result":

(1) Form the denominator and the numerator of the network function in polynomials.

(2) Reduce the degrees of "S" if there are common factors in the numerator and denominator.

(3) Print out the network functions.

Block G

Print out: program time

If more than one problem is inserted in the program.

4.6 Topological Formulas.

The topological formulas [6]-[8] used in this program are listed below:

$$Y_{dp} = \frac{V(Y)}{W_{1,1'}(Y)} \qquad (4\text{-}6\text{-}1)$$

$$Z_{11} = \frac{W_{1,1'}(Y)}{V(Y)} \qquad (4\text{-}6\text{-}2)$$

$$Z_{12}=Z_{21} = \frac{\frac{1}{2}\left[W_{1,2'}(Y) + W_{2,1'}(Y) - W_{1,2}(Y) - W_{1',2'}(Y)\right]}{V(Y)} \qquad (4\text{-}6\text{-}3)$$

$$Z_{22} = \frac{W_{2,2'}(Y)}{V(Y)} \qquad (4\text{-}6\text{-}4)$$

$$Y_{11} = \frac{W_{2,2'}(Y)}{\Sigma U} \qquad (4\text{-}6\text{-}5)$$

$$Y_{12}=Y_{21} = \frac{\frac{1}{2}\left[W_{1,2}(Y) + W_{1',2'}(Y) - W_{1,2'}(Y) - W_{2,1'}(Y)\right]}{\Sigma U} \qquad (4\text{-}6\text{-}6)$$

$$Y_{22} = \frac{W_{1,1'}(Y)}{\Sigma U} \qquad (4\text{-}6\text{-}7)$$

$$\mu_{12} = \frac{\frac{1}{2}\left[W_{1,2'}(Y) + W_{2,1'}(Y) - W_{1,2}(Y) - W_{1',2'}(Y)\right]}{W_{1,1'}(Y)} \qquad (4\text{-}6\text{-}8)$$

$$\mu_{21} = \frac{\frac{1}{2}\left[W_{1,2'}(Y) + W_{2,1'}(Y) - W_{1,2}(Y) - W_{1',2'}(Y)\right]}{W_{2,2'}(Y)} \qquad (4\text{-}6\text{-}9)$$

$$\alpha_{12} = \frac{\frac{1}{2}\left[W_{1,2}(Y) + W_{1',2'}(Y) - W_{1,2'}(Y) - W_{2,1'}(Y)\right]}{W_{2,2'}(Y)} \qquad (4\text{-}6\text{-}10)$$

$$\alpha_{21} = \frac{\frac{1}{2}\left[W_{1,2}(Y) + W_{1',2'}(Y) - W_{1,2'}(Y) - W_{2,1'}(Y)\right]}{W_{1,1'}(Y)} \qquad (4\text{-}6\text{-}11)$$

5. Conclusion

The results of the two new tree-finding programs show that the computation times and consequently the execution times for the determination of the network functions are greatly reduced compared to other methods. Furthermore, the program for determining network functions in Appendix III can be modified for generalized network analysis and synthesis.

BIBLIOGRAPHY

1. W. Mayada and S. Seshu, "Topological Formulas for Network Functions," Univ. of Illinois, Bul. Vol. 55, No. 23, Urbana, Ill., November 1957.

2. Weinberg, "Kirchhoff's Third and Fourth Law," IRE Transactions on Circuit Theory, CT-5, March 1958.

3. S. P. Chan, "Introductory Topological Analysis of Electrical Networks," Hold, Rinehart and Winston, 1968.

4. F. Reza, "Some Topological Consideration in Networks Theory," CT-5 IRE, Transactions on Circuit Theory, 1958.

5. S. Seshu and M. B. Reed, "Linear Graphs and Electrical Networks," Addison Wesley, 1961.

6. S. P. Chan and S. G. Chan, "Topological Formula for Digital Computation," Proceeding First Annual Princeton Conference on Information Sciences and Systems, 1967, pp. 906-912.

7. France Rode and S. P. Chan, "Computer Evaluation of Topological Formulas for Network Analysis," Proceeding of the Hawaii International Conference on System Sciences, p. 796, 1968.

8. S. P. Chan and Shu-Gar Chan, "Modification of Topological Formulas," IEEE Transaction on Circuit Theory, CT-15, March 1968.

9. Wai-Kai Chen, "Generation of Trees and K-Trees," Proc. of Third Allerton Conference on Circuit and System Theory, pp. 906-912, October 1965.

10. L. M. Maxwell and J. M. Cline, "Topological Network Analysis by Algebraic Methods," Proc. IEE (Great Britain) Vol. 113, No. 8, August 1966.

11. W. Mayeda and S. Seshu, "Generation of Trees Without Duplications," IEEE Transaction on Circuit Theory, Vol. CT-12, No. 12, p. 18, June 1965.

12. G. J. Minty, "A Simple Algorithm for Listing All the Trees of a Graph," IEEE Transaction on Circuit Theory, March 1965, CT-12, p. 120.

13. B. R. Myers and L. V. Auth, Jr., "The Number and Listing of All Trees in an Arbitrary Graph," Technical Report EE-6511, 1965, University of Notre Dame, Indiana.

14. M. Piekarsky, "Listing of All Possible Trees of a Linear Graph," IEEE Transaction on Circuit Theory, March 1965, CT-12, p. 124.

15. G. W. Zobrist and G. V. Lago, "Digital Computer Analysis of Passive Networks Using Topological Formulas," The University of Missouri Bulletin, Vol. 66, No. 12, Engineering Experiment Station Series No. 59, March 29, 1965.

APPENDIX I

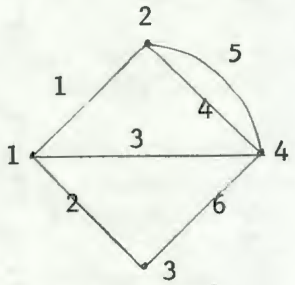
The digital computer program for finding trees by using the dichotomy method, as described in section 3.1, is presented below.

Language: FORTRAN IV

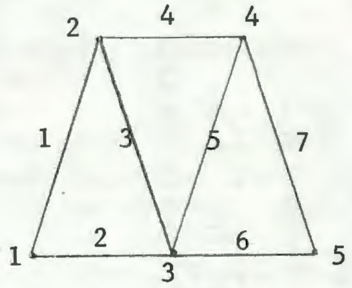
Machine: IBM 360/67

Problems: 7

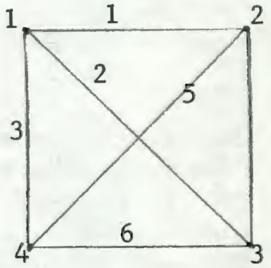
The linear graphs corresponding to each problem are shown below.



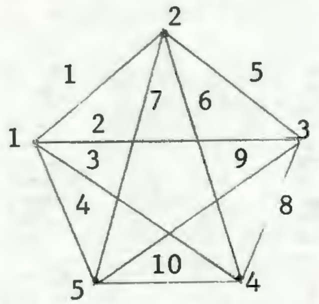
(a) Problem 1



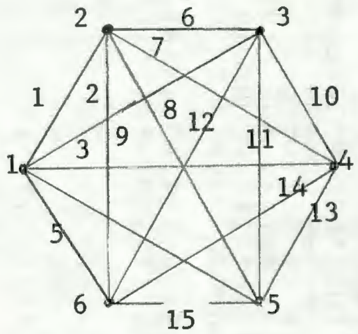
(b) Problem 2



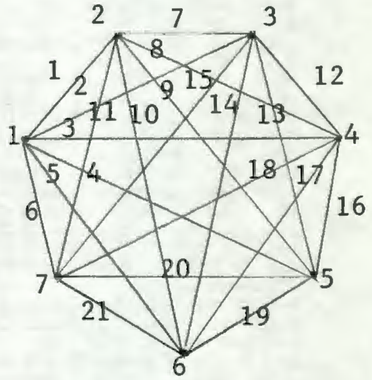
(c) Problem 3



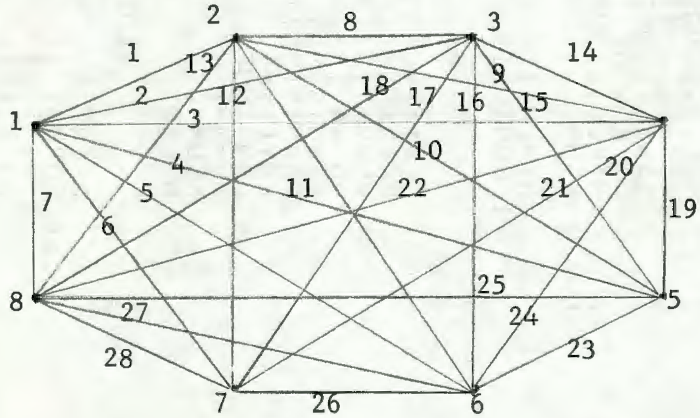
(d) Problem 4



(e) Problem 5



(f) Problem 6



(g) Problem 7

```
C       THIS IS A TREE FINDING PROGRAM BY DICHOTOMY METHOD
C       THE INPUTS ARE NO. OF VERTEX,NO. OF EDGES AND THE
C          INCIDENCE MATRIX OF A GRAPH
C       MORE THAN ONE GRAPHS CAN BE SOLVED WITH THE SAME
C          PROGRAM AT A TIME
C       N......THE NUMBER OF VERTICES OF A GIVEN GRAPH
C       MB.....THE NUMBER OF EDGES OF A GIVEN GRAPH
C       LIST...CONDITIONS FOR TREE PRINTING
C          LIST=1...PRINT ALL THE TREES THAT HAVE BEEN FOUND
C          LIST=0...DO NOT PRINT THEM OUT AT ALL
C       KAM.....INCIDENCE MATRIX OF A GIVEN LINEAR GRAPH
C       KS......MATRIX DESIGNATED FOR OPERATING VERTEX
C       KA......MATRIX DESIGNATED FOR BEING COMPARISON
C       CL......CLOCK TIME
C       NOL.....NO. OF THE 'GROUP OF COMMON SETS'
C       MTREE...MATRIX OF THE COMMON SETS
C       IANS....ONE TREE BY TAKING ONE ELEMENT FROM EACH
C          ROW OF 'MTREE' MATRIX,THE COMMON SETS
C       NTR.....NUMBER OF TREES FOUND IN EACH 'MTRIX'MATRIX
C       KSUM....TOTAL NUMBER HAVE BEEN FOUND OF THE GRAPH
C
        DIMENSION KEY(15),KS(15,60),KAM(15,60),LTMP(15,60),
       1 CONTL(15), MTREE(15,60),KA(15,15,60)
C
C              BLOCK  A
C
  1100 CL=ITIME(0)*.01
        READ(5,55,END=1111) N,MB,LIST
    55 FORMAT(3I10)
        DO 44 I=1,N
        READ(5,50)  (KAM(I,J),J=1,MB)
    50 FORMAT(30I1)
        KEY(I)=0
    44 CONTINUE
        NV=N-1
        KSUM=0
        NOL=0
C
C              BLOCK  B
C
        PRINT 66, N, MB ,LIST
    66 FORMAT (1H1,8X,'NODES=',I3,', EDGES='I3,', PRINT CONDITION='I2//,
       18X,'INCIDENCE MATRIX'//)
        DO 2 I=1,N
        PRINT  51, (  KAM(I,J),J=1,MB)
```

```
   51 FORMAT (7X,30I2)
    2 CONTINUE
      IF (LIST) 115,116,115
  115 PRINT 68
   68 FORMAT(/10X,' TREES'/)
      GO TO 70
  116 PRINT 69
   69 FORMAT(/7X,'TREES HAVE BEEN FOUND,BUT NOT PRINTED')
C
C           BLOCK  C
C
   70 DO 59 I=1,NV
      DO 59 J=1,MB
   59 KA(1,I,J)=KAM(I+1,J)
      DO 60 J=1,MB
   60 KS(1,J)=KAM(1,J)
      NX1=N-1
      NX2=N-2
      NX3=N-3
      NX4=N-4
      NX5=N-5
      NX6=N-6
      NX7=N-7
      NX8=N-8
      NX9=N-9
      NX10=N-10
      NX11=N-11
      NX12=N-12
      NX13=N-13
      NX14=N-14
C
C           BLOCK  D
C
      CONTL(1)=1.0
      DO 100 I1=1,NX1
      NX=NX1
      M=1
      IF (CONTL(M).EQ.0.0) GO TO 300
      CALL       NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      IF (NX2.EQ.0) GO TO 190
      IF (CONTL(M+1).EQ.0.0) GO TO 300
      DO  99 I2=1,NX2
      NX=NX2
      M=2
      IF (CONTL(M).EQ.0.0) GO TO 100
```

```
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX3.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 100
DO  98 I3=1,NX3
NX=NX3
M=3
IF (CONTL(M).EQ.0.0) GO TO  99
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX4.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 99
DO  97 I4=1,NX4
NX=NX4
M=4
IF (CONTL(M).EQ.0.0) GO TO 98
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX5.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 98
DO 96 I5 =1,NX5
NX=NX5
M=5
IF (CONTL(M).EQ.0.0) GO TO 97
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX6.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 97
DO 95 I6 =1,NX6
NX=NX6
M=6
IF (CONTL(M).EQ.0.0) GO TO 96
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX7.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 96
DO 94 I7 =1,NX7
M=7
IF (CONTL(M).EQ.0.0) GO TO 95
NX=NX7
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX8.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 95
DO 93 I8 =1,NX8
NX=NX8
M=8
IF (CONTL(M).EQ.0.0) GO TO 94
CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
IF (NX9.EQ.0) GO TO 190
IF (CONTL(M+1).EQ.0.0) GO TO 94
```

```
      DO 92 I9 =1,NX9
      NX=NX9
      M=9
      IF (CONTL(M).EQ.0.0) GO TO 93
      CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      IF(NX10.EQ.0) GO TO 190
      IF (CONTL(M+1).EQ.0.0) GO TO 93
      DO 91 I10=1,NX10
      NX=NX10
      M=10
      IF (CONTL(M).EQ.0.0) GO TO 92
      CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      IF(NX11.EQ.0) GO TO 190
      IF (CONTL(M+1).EQ.0.0) GO TO 92
      DO 90 I11=1,NX11
      NX=NX11
      M=11
      IF (CONTL(M).EQ.0.0) GO TO 91
      CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      IF(NX12.EQ.0) GO TO 190
      IF (CONTL(M+1).EQ.0.0) GO TO 91
      DO 89 I12=1,NX12
      NX=NX12
      M=12
      IF (CONTL(M).EQ.0.0) GO TO 90
      CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      IF(NX13.EQ.0) GO TO 190
      IF (CONTL(M+1).EQ.0.0) GO TO 90
      DO 88 I13=1,NX13
      NX=NX13
      M=13
      IF (CONTL(M).EQ.0.0) GO TO 89
      CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      IF(NX14.EQ.0) GO TO 190
      IF (CONTL(M+1).EQ.0.0) GO TO 89
      DO 87 I14=1,NX14
      NX=NX14
      M=14
      IF (CONTL(M).EQ.0.0) GO TO 88
      CALL        NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
190   CONTINUE
      NOL=NOL+1
      NTR=1
      DO 105 I=1,NV
105   NTR=NTR*KEY(I)
```

```
      KSUM=KSUM+NTR
  216 CONTINUE
C         THE 'MTREE' MATRIX CAN BE PRINTED HERE
      CALL          TREE (KEY,MTREE,LIST)
      GO TO (100,99,98,97,96,95,94,93,92,91,90,89,88,87),M
   87 CONTINUE
   88 CONTINUE
   89 CONTINUE
   90 CONTINUE
   91 CONTINUE
   92 CONTINUE
   93 CONTINUE
   94 CONTINUE
   95 CONTINUE
   96 CONTINUE
   97 CONTINUE
   98 CONTINUE
   99 CONTINUE
  100 CONTINUE
C
C                BLOCK   E
C
  300 PRINT 251,KSUM
  251 FORMAT (/8X,'TOTAL NO. OF TREES=',I7)
      CL=ITIME(0)*.01-CL
      WRITE(6,304) CL
  304 FORMAT (//7X,' TIME= ',F7.2,' SECONDS')
      GO TO 1100
 1111 STOP
      END



      SUBROUTINE NPART(M,MB,KEY,KS,KA,LTMP,CONTL,NX,MTREE)
      DIMENSION KEY(15),KS(15,60),KA(15,15,60),LTMP(15,60),
     1 CONTL(15),MTREE(15,60)
      KEY(M)=0
      KEND=0
      KCPR=0
      NTEST=0
      MJ=0
      DO 2  J=1,MB
      K=J
      IF (KS(M,J).EQ.0) GO TO 2
      GO TO 9
    2 CONTINUE
    9 DO 3 I=1,NX
      IK=I
      IF(KA(M,I,K).EQ.0)GO TO 3
      GO TO 15
    3 CONTINUE
   15 DO 72 J=1,MB
      LTMP(M,J)=KS(M,J)*KA(M,IK,J)
      KEY(M)=KEY(M)+LTMP(M,J)
      KA(M,IK,J)=KA(M,IK,J)-LTMP(M,J)
      KS(M,J)=KS(M,J)-LTMP(M,J)
      KS(M+1,J)=KS(M,J)+KA(M,IK,J)
      KEND=KEND+KS(M,J)
      KCPR=KCPR+KA(M,IK,J)
      NTEST=NTEST+KS(M+1,J)
      IF(LTMP(M,J).EQ.0) GO TO 72
      MJ=MJ+1
      MTREE(M,MJ)=J
   72 CONTINUE
      DO 12 J=1,MB
      ITEMP=KA(M,NX,J)
      KA(M,NX,J)=KA(M,IK,J)
   12 KA(M,IK,J)=ITEMP
      K=NX-1
      DO 66 I=1,K
      DO 66 J=1,MB
   66 KA(M+1,I,J)=KA(M,I,J)
      CONTL(M)=KEND*KCPR
      CONTL(M+1)=NTEST
      RETURN
      END
```

```
      SUBROUTINE  TREE (KEY,MTREE,LIST)
      DIMENSION  KEY(15),MTREE(15,60),IANS(15)
      L1=KEY(1)
      DO 21 I1=1,L1
      M=1
      IANS(M)=MTREE(M,I1)
      IF(KEY(M+1).EQ.0) GO TO 200
```

```
L2=KEY(M+1)
DO 22 I2=1,L2
M=2
IANS(M)=MTREE(M,I2)
IF(KEY(M+1).EQ.0) GO TO 200
L3=KEY(M+1)
DO 23 I3=1,L3
M=3
IANS(M)=MTREE(M,I3)
IF(KEY(M+1).EQ.0) GO TO 200
L4=KEY(M+1)
DO 24 I4=1,L4
M=4
IANS(M)=MTREE(M,I4)
IF(KEY(M+1).EQ.0) GO TO 200
L5=KEY(M+1)
DO 25 I5=1,L5
M=5
IANS(M)=MTREE(M,I5)
IF(KEY(M+1).EQ.0) GO TO 200
L6=KEY(M+1)
DO 26 I6=1,L6
M=6
IANS(M)=MTREE(M,I6)
IF(KEY(M+1).EQ.0) GO TO 200
L7=KEY(M+1)
DO 27 I7=1,L7
M=7
IANS(M)=MTREE(M,I7)
IF(KEY(M+1).EQ.0) GO TO 200
L8 =KEY(M+1)
DO 28 I8=1,L8
M=8
IANS(M)=MTREE(M,I8)
IF(KEY(M+1).EQ.0) GO TO 200
L9=KEY(M+1)
DO 29 I9=1,L9
M=9
IANS(M)=MTREE(M,I9)
IF(KEY(M+1).EQ.0) GO TO 200
L10=KEY(M+1)
DO 30 I10=1,L10
M=10
IANS(M)=MTREE(M,I10)
IF(KEY(M+1).EQ.0) GO TO 200
```

```
      L11=KEY(M+1)
      DO 31 I11=1,L11
      M=11
      IANS(M)=MTREE(M,I11)
      IF(KEY(M+1).EQ.0) GO TO 200
      L12=KEY(M+1)
      DO 32 I12=1,L12
      M=12
      IANS(M)=MTREE(M,I12)
      IF(KEY(M+1).EQ.0) GO TO 200
      L13=KEY(M+1)
      DO 33 I13=1,L13
      M=13
      IANS(M)=MTREE(M,I13)
      IF(KEY(M+1).EQ.0) GO TO 200
      L14=KEY(M+1)
  200 IF (LIST.EQ.0) GO TO 100
      PRINT 50,(IANS(I),I=1,M)
   50 FORMAT (10X,15I3)
  100 GO TO (21,22,23,24,25,26,27,28,29,30,31,32,33,34),M
      DO 34 I14=1,L14
      M=14
      IANS(M)=MTREE(M,I14)
   34 CONTINUE
   33 CONTINUE
   32 CONTINUE
   31 CONTINUE
   30 CONTINUE
   29 CONTINUE
   28 CONTINUE
   27 CONTINUE
   26 CONTINUE
   25 CONTINUE
   24 CONTINUE
   23 CONTINUE
   22 CONTINUE
   21 CONTINUE
      RETURN
      END
```

```
 NODES=  4, EDGES=  6, PRINT CONDITION= 1
 INCIDENCE MATRIX
1 1 1 0 0 0
1 0 0 1 1 0
0 1 0 0 0 1
0 0 1 1 1 1
  TREES
    1  2  3
    1  2  4
    1  2  5
    1  2  6
    1  3  6
    1  4  6
    1  5  6
    2  3  4
    2  3  5
    2  6  4
    2  6  5
    3  4  6
    3  5  6
 TOTAL NO. OF TREES=      13
 TIME=     0.63 SECONDS
 NODES=  5, EDGES=  7, PRINT CONDITION= 1
 INCIDENCE MATRIX
1 1 0 0 0 0 0
1 0 1 1 0 0 0
0 1 1 0 1 1 0
0 0 0 1 1 0 1
0 0 0 0 0 1 1
  TREES
   1  2  4  6
   1  2  4  7
   1  2  5  6
   1  2  5  7
   1  3  4  6
   1  3  4  7
   1  3  5  6
   1  3  5  7
   1  2  6  7
   1  3  6  7
   1  4  5  6
   1  4  5  7
   1  4  7  6
   2  3  4  6
   2  3  4  7
   2  3  5  6
   2  3  5  7
   2  3  6  7
   2  5  4  6
   2  5  4  7
   2  6  7  4
TOTAL NO. OF TREES=       21
TIME=     0.13 SECONDS
```

```
 NODES=  4, EDGES=  6, PRINT CONDITION= 0
 INCIDENCE MATRIX
1 1 1 0 0 0
1 0 0 1 1 0
0 1 0 1 0 1
0 0 1 0 1 1
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=     16
 TIME=    0.06 SECONDS
```

```
 NODES=  5, EDGES= 10, PRINT CONDITION= 0
 INCIDENCE MATRIX
1 1 1 1 0 0 0 0 0 0
1 0 0 0 1 1 1 0 0 0
0 1 0 0 0 0 1 1 1 0
0 0 1 0 0 1 0 0 1 1
0 0 0 1 1 0 0 1 0 1
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=    125
 TIME=    0.25 SECONDS
 NODES=  6, EDGES= 15, PRINT CONDITION= 0
 INCIDENCE MATRIX
1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 1 0 0 0 1 0 0 0 1 1 1 0 0 0
0 0 1 0 0 0 1 0 0 1 0 0 1 1 0
0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 0 0 1 0 0 0 1 0 0 1 0 1 1
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=    1296
 TIME=     1.56 SECONDS
```

```
 NODES=  7, EDGES= 21, PRINT CONDITION= 0
 INCIDENCE MATRIX
1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0
0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0
0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=  16807
 TIME=   12.75 SECONDS
 NODES=  8, EDGES= 28, PRINT CONDITION= 0

 INCIDENCE MATRIX

1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 1 0
0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 0 1 1
TREES HAVE BEEN FOUND, BUT NOT PRINTED

 TOTAL NO. OF TREES= 262144

 TIME=  124.50 SECONDS
```

APPENDIX II

The digital computer program for finding trees by using the T-triangle method, as described in section 3.3, is presented below.

Language: FORTRAN IV
Machine: IBM 360/67
Problems: 7

The linear graphs corresponding to each problem are the same as shown in Appendix I.

```
C     THIS IS A TREE FINDING PROGRAM BY T-TRIANGLE METHOD
C     THE INPUTS ARE NO. OF VERTEX,NO. OF EDGES AND THE
C        'LINK TRIANGULAR ARRAY'
C     MORE THAN ONE GRAPHS CAN BE SOLVED WITH THE SAME
C        PROGRAM AT A TIME
C     N......THE NUMBER OF VERTICES OF A GIVEN GRAPH
C     MB.....THE NUMBER OF EDGES OF A GIVEN GRAPH
C     LIST...CONDITIONS FOR TREE PRINTING
C        LIST=1...PRINT ALL THE TREES THAT HAVE BEEN FOUND
C        LIST=0...DO NOT PRINT THEM OUT AT ALL
C     NT......T-TRIANGLE,IN 3 DIMENSION
C     IPP.....ONE DIMNNSION MATRIX,NO. OFEDGES BETWEEN
C        EACH PAIR OF VERTICES
C     ITREE...ONE TREE BY TAKING ONE ELEMENT FROM EACH
C        ROW OF 'MTREE' MATRIX,THE COMMON SETS
C     NTR.....NUMBER OF TREES FOUND IN EACH 'MTREE'MATRIX
C     KSUM....TOTAL NUMBER HAVE BEEN FOUND OF THE GRAPH
C
      DIMENSION NT(25,25,25),IROW(25),ICOL(25),MTREE(25,50),
     1 IND(25),LINK(25,25),IPP(300),KEY(25)
 1100 CL=ITIME(0)*.01
C
C             BLOCK  A
C
      READ (5,45,END=1111) N,MB,LIST
   45 FORMAT (3I10)
      NV=N-1
      DO 50 I=1,NV
   50 READ (5,60) (LINK(I,J),J=I,NV)
   60 FORMAT (30I1)
      KF=LINK(1,1)
      DO 51 I=1,NV
      DO 51 J=I,NV
      IF (LINK(I,J).EQ.0) GO TO 49
      NT(1,J,I)=KF
      IPP(KF)=LINK(I,J)
      KF=NT(1,J,I)+LINK(I,J)
      GO TO 51
   49 NT(1,J,I)=0
   51 CONTINUE
C
C             BLOCK  B
C
      PRINT 105,N,MB,LIST
  105 FORMAT (1H1,8X,'NODES=',I3,', EDGES='I3,', PRINT CONDITION='I2//)
```

```
      KF=KF-LINK(I,J)
      PRINT 62,KF
   62 FORMAT (8X,'EDGES CALCULATED='I3//10X,'INPUT LINK'//)
      DO 63 I=1,NV
   63 PRINT 67,(LINK(I,J),J=I,NV)
      PRINT 64
   64 FORMAT (/'          INPUT T-TRIANGLE'//)
      DO 65 I=1,NV
   65 PRINT 67,(NT(1,I,J),J=1,I)
   67 FORMAT (10X,25I3/)
      IF (LIST) 115,116,115
  115 PRINT 68
   68 FORMAT(/10X,' TREES'/)
      GO TO 70
  116 PRINT 69
   69 FORMAT(/7X,'TREES HAVE BEEN FOUND,BUT NOT PRINTED')
   70 NO=0
      NSUM=0
      N1=N-1
      N2=N-2
      N3=N-3
      N4=N-4
      N5=N-5
      N6=N-6
      N7=N-7
      N8=N-8
      N9=N-9
      N10=N-10
      N11=N-11
      N12=N-12
      N13=N-13
      N14=N-14
C
C           BLOCK  C
C
      K=1
      DO 1 I1=1,N1
      M=1
      IX=I1
      IND(M)=M
      CALL       CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
      IF (KEY(M).EQ.0) GO TO 34
      IF (M.EQ.NV) GO TO 200
      DO 2 I2=1,N2
      M=2
```

```
IX=I2
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL         CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 33
IF (M.EQ.NV) GO TO 200
DO 3 I3 =1,N3
M=3
IX=I3
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL         CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 32
IF (M.EQ.NV) GO TO 200
DO 4 I4 =1,N4
M=4
IX=I4
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL         CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 31
IF (M.EQ.NV) GO TO 200
DO 5 I5 =1,N5
M=5
IX=I5
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL         CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 30
IF (M.EQ.NV) GO TO 200
DO 6 I6 =1,N6
M=6
IX=I6
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL         CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 29
IF (M.EQ.NV) GO TO 200
DO 7 I7 =1,N7
M=7
IX=I7
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL         CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 28
IF (M.EQ.NV) GO TO 200
DO  8 I8 =1,N8
M=8
IX=I8
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 27
IF (M.EQ.NV) GO TO 200
DO  9 I9 =1,N9
M=9
IX=I9
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 26
IF (M.EQ.NV) GO TO 200
M=10
DO 10 I10=1,N10
IX=I10
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 25
IF (M.EQ.NV) GO TO 200
DO 11 I11=1,N11
M=11
IX=I11
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 24
IF (M.EQ.NV) GO TO 200
DO 12 I12=1,N12
M=12
IX=I12
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
IF (KEY(M).EQ.0) GO TO 23
IF (M.EQ.NV) GO TO 200
DO 13 I13=1,N13
M=13
IX=I13
IND(M)=M
```

```
      IF (IX.EQ.1)   IND(M)=IND(M-1)
      CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
      IF (KEY(M).EQ.0) GO TO 22
      IF (M.EQ.NV) GO TO 200
      DO 14 I14=1,N14
      M=14
      IX=I14
      IND(M)=M
      IF (IX.EQ.1)   IND(M)=IND(M-1)
      CALL        CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
C
C           BLOCK  D
C
  200 NO=NO+1
      NTR=1
      DO 15 I=1,NV
   15 NTR=NTR*KEY(I)
      NSUM=NSUM+NTR
C           PRINT T- MATRIX AT HERE
      CALL         TREE(KEY,MTREE,NV,LIST)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14),NV
   14 CONTINUE
   22 JX=14
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 23
      INX=IND(JX-1)
      CALL         NEWTRI(INX,JX,N,NV,NT,K)
   13 CONTINUE
   23 JX=13
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 24
      INX=IND(JX-1)
      CALL         NEWTRI(INX,JX,N,NV,NT,K)
   12 CONTINUE
   24 JX=12
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 25
      INX=IND(JX-1)
      CALL         NEWTRI(INX,JX,N,NV,NT,K)
   11 CONTINUE
   25 JX=11
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 26
      INX=IND(JX-1)
      CALL         NEWTRI(INX,JX,N,NV,NT,K)
   10 CONTINUE
   26 JX=10
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 27
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    9 CONTINUE
   27 JX=9
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 28
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    8 CONTINUE
   28 JX=8
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 29
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    7 CONTINUE
   29 JX=7
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 30
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    6 CONTINUE
   30 JX=6
      INX=IND(JX-1)
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 31
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    5 CONTINUE
   31 JX=5
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 32
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    4 CONTINUE
   32 JX=4
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 33
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    3 CONTINUE
   33 JX=3
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 34
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    2 CONTINUE
   34 JX=2
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 35
      INX=IND(JX-1)
      CALL        NEWTRI(INX,JX,N,NV,NT,K)
    1 CONTINUE
C
C                 BLOCK E
C
```

```
   35 CL=ITIME(0)*.01-CL
      PRINT  301,NSUM
  301 FORMAT (/8X,'TOTAL NO. OF TREES=',I7)
      WRITE(6,300) CL
  300 FORMAT (//7X,' TIME= ',F7.2,' SECONDS')
      GO TO 1100
 1111 STOP
      END



      SUBROUTINE CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,IPP)
      DIMENSION NT(25,25,25),IROW(25),ICOL(25),MTREE(25,50),
     1 IPP(300),KEY(25)
      MQ=M+1
      IROW(MQ)=0
      ICOL(MQ)=0
      IF ((M+IX).GT.NV) GO TO 12
      DO 11 J =MQ,NV
      ICOL(MQ)=NT(K,J,MQ)+ICOL(MQ)
      DO 11 I =1,M
   11 IROW(MQ)=NT(K,J,I)+IROW(MQ)
   12 JJ=0
      DO 16 I =1,M
      IF (NT(K,M,I).EQ.0) GO TO 16
      KF=NT(K,M,I)
      JP=IPP(KF)
      DO 18 J=1,JP
      JJ=JJ+1
   18 MTREE(M,JJ)=KF+J-1
   16 CONTINUE
      KEY(M)=JJ
      RETURN
      END



      SUBROUTINE  NEWTRI(INX,JX,N,NV,NT,K)
      DIMENSION  NT(25,25,25),ITEMP(25)
   19 NUM=JX-1
      K=JX-1
      DO 20 I =1,NUM
   20 ITEMP(I)=0
      NJ=N-JX
      DO 21 I=JX,NV
      ITEMP(I)=NT(INX,I,JX)
   21 CONTINUE
C           MOVE THE ENTIES ON BOTH SIDES OF THE OPERATING
C           COLUMN JX ACCORDING TO THE ALGORITHM
      DO 22 I =1,NJ
      L=JX-2+I
      DO 24 J =1,L
      IF(J.GE.JX) GO TO 23
      NT(K,L,J)=NT(INX,L+1,J)
      GO TO 24
   23 NT(K,L,J)=NT(INX,L+1,J+1)
   24 CONTINUE
   22 CONTINUE
      DO 25 J =1,NV
   25 NT(K,NV,J)=ITEMP(J)
   35 RETURN
      END
```

```
      SUBROUTINE  TREE(KEY,MTREE,NV,LIST)
      DIMENSION KEY(25),MTREE(25,50),ITREE(25)
      N1=KEY(1)
      DO 1 I1=1,N1
      K=1
      ITREE(K)=MTREE(K,I1 )
      IF (K.EQ.NV) GO TO 200
      N2=KEY(K+1)
      DO 2 I2=1,N2
      K=2
      ITREE(K)=MTREE(K,I2 )
      IF (K.EQ.NV) GO TO 200
      N3=KEY(K+1)
      DO 3 I3 =1,N3
      K=3
      ITREE(K)=MTREE(K,I3 )
      IF (K.EQ.NV) GO TO 200
      N4=KEY(K+1)
      DO 4 I4 =1,N4
```

```
K=4
ITREE(K)=MTREE(K,I4 )
IF (K.EQ.NV) GO TO 200
N5=KEY(K+1)
DO 5 I5 =1,N5
K=5
ITREE(K)=MTREE(K,I5 )
IF (K.EQ.NV) GO TO 200
N6=KEY(K+1)
DO 6 I6 =1,N6
K=6
ITREE(K)=MTREE(K,I6 )
IF (K.EQ.NV) GO TO 200
N7=KEY(K+1)
DO 7 I7=1,N7
K=7
ITREE(K)=MTREE(K,I7 )
IF (K.EQ.NV) GO TO 200
N8=KEY(K+1)
DO 8 I8=1,N8
K=8
ITREE(K)=MTREE(K,I8 )
IF (K.EQ.NV) GO TO 200
N9=KEY(K+1)
DO 9 I9=1,N9
K=9
ITREE(K)=MTREE(K,I9 )
IF (K.EQ.NV) GO TO 200
N10=KEY(K+1)
DO 10 I10=1,N10
K=10
ITREE(K)=MTREE(K,I10)
IF (K.EQ.NV) GO TO 200
N11=KEY(K+1)
DO 11 I11=1,N11
K=11
ITREE(K)=MTREE(K,I11)
IF (K.EQ.NV) GO TO 200
N12=KEY(K+1)
DO 12 I12=1,N12
K=12
ITREE(K)=MTREE(K,I12)
IF (K.EQ.NV) GO TO 200
N13=KEY(K+1)
DO 13 I13=1,N13
```

```
      K=13
      ITREE(K)=MTREE(K,I13)
      IF (K.EQ.NV) GO TO 200
      N14=KEY(K+1)
      DO 14 I14=1,N14
      K=14
      ITREE(K)=MTREE(K,I14)
  200 CONTINUE
C           STATEMENT OF PRINT ONE TREE IS INSERTED HERE
      IF (LIST.EQ.0) GO TO 100
      PRINT 50,(ITREE(I),I=1,NV)
   50 FORMAT (10X,25I3)
  100 GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14),NV
   14 CONTINUE
   13 CONTINUE
   12 CONTINUE
   11 CONTINUE
   10 CONTINUE
    9 CONTINUE
    8 CONTINUE
    7 CONTINUE
    6 CONTINUE
    5 CONTINUE
    4 CONTINUE
    3 CONTINUE
    2 CONTINUE
    1 CONTINUE
      RETURN
      END
```

```
NODES=  4, EDGES=  6, PRINT CONDITION= 1

EDGES CALCULATED=  6

  INPUT LINK

  1  1  1
  0  2
  1

 INPUT T-TRIANGLE

  1
  2  0
  3  4  6

 TREES

  1  2  3
  1  2  4
  1  2  5
  1  2  6
  1  3  6
  1  4  6
  1  5  6
  2  3  4
  2  3  5
  2  6  4
  2  6  5
  3  4  6
  3  5  6

TOTAL NO. OF TREES=      13

TIME=     0.63 SECONDS
```

```
NODES=  5, EDGES=  7, PRINT CONDITION= 1
EDGES CALCULATED=  7
  INPUT LINK

  1  1  0  0
  1  1  0
  1  1
  1

 INPUT T-TRIANGLE

  1
  2  3
  0  4  5
  0  0  6  7

 TREES

  1  2  4  6
  1  2  4  7
  1  2  5  6
  1  2  5  7
  1  3  4  6
  1  3  4  7
  1  3  5  6
  1  3  5  7
  1  2  6  7
  1  3  6  7
  1  4  7  5
  1  4  7  6
  1  4  5  6
  2  5  6  3
  2  5  6  4
  2  5  7  3
  2  5  7  4
  2  6  3  7
  2  6  3  4
  2  6  7  4
  2  3  4  7
TOTAL NO. OF TREES=     21
TIME=    0.06 SECONDS
```

```
 NODES=  4, EDGES=  6, PRINT CONDITION= 0
 EDGES CALCULATED=  6
   INPUT LINK

   1  1  1
   1  1
   1
  INPUT T-TRIANGLE

   1
   2  4
   3  5  6
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=      16
 TIME=    0.06 SECONDS
 NODES=  5, EDGES= 10, PRINT CONDITION= 0
 EDGES CALCULATED= 10
   INPUT LINK
   1  1  1  1
   1  1  1
   1  1
   1
  INPUT T-TRIANGLE
   1
   2  5
   3  6  8
   4  7  9 10
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=    125
 TIME=    0.13 SECONDS
```

```
NODES=  6, EDGES= 15, PRINT CONDITION= 0
EDGES CALCULATED= 15
  INPUT LINK
  1  1  1  1  1
  1  1  1  1
  1  1  1
  1  1
  1
 INPUT T-TRIANGLE
  1
  2  6
  3  7 10
  4  8 11 13
  5  9 12 14 15
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=   1296
 TIME=     0.38 SECONDS
```

```
NODES=  7, EDGES= 21, PRINT CONDITION= 0
EDGES CALCULATED= 21
  INPUT LINK
  1  1  1  1  1  1
  1  1  1  1  1
  1  1  1  1
  1  1  1
  1  1
  1
 INPUT T-TRIANGLE
  1
  2  7
  3  8 12
  4  9 13 16
  5 10 14 17 19
  6 11 15 18 20 21
TREES HAVE BEEN FOUND, BUT NOT PRINTED
 TOTAL NO. OF TREES=  16807
 TIME=    3.00 SECONDS
```

```
NODES=  8, EDGES= 28, PRINT CONDITION= 0

EDGES CALCULATED= 28

  INPUT LINK

  1  1  1  1  1  1  1
  1  1  1  1  1  1
  1  1  1  1  1
  1  1  1  1
  1  1  1
  1  1
  1

 INPUT T-TRIANGLE

  1
  2  8
  3  9 14
  4 10 15 19
  5 11 16 20 23
  6 12 17 21 24 26
  7 13 18 22 25 27 28
TREES HAVE BEEN FOUND, BUT NOT PRINTED

TOTAL NO. OF TREES= 262144

TIME=   32.87 SECONDS
```

APPENDIX III

A computer program for the determination of network functions. The same networks as given in examples2.1 and 2.3 are solved in this program.

Language: FORTRAN IV
Machine: IBM 360/67
Problems: 2

```
C     THIS PROGRAM IS FOR THE DETERMINATION OF NETWORK
C        FUNTIONS BY USING TOPOLOGICAL FORMULAS
C        AT A TIME
C     N.....THE NUMBER OF VERTICIES OF A GIVEN NETWORK
C     MB....THE NUMBER OF EDGES OF A GIVEN NETWORK
C     IGR...GROUD CONDITION
C        IGR=1.... THE VERTICES 1' AND 2'OF A 2-P NETWORK
C                  ARE SHORTED,AND DESIGNATED AS VERTEX 3.
C        IGR=2.... THE VERTICES 1' AND 2' OF A 2-P NETWORK
C                  ARE NOT SHORTED,AND DESIGNATED AS VERTEX
C                3 AND VERTEX 4 RESPECTIVELY IN THIS PROGRAM
C        NO CONSTRAIN FOR 1-P NETWOEK
C     IPORT..PORT CONDITION
C        IPORT=1  FOR 1-P
C        IPORT=2  FOR 2-P
C     LE.....LABELLING NUMBER OF  A EDGE
C        CONNECTED BY LE , NX MUST BE LESS THAN NY
C     NK......ORIGINAL T-TRIANGLE,IN MATRIX FORM
      INTEGER A,DASH
      DIMENSION NK(20,25),IPP(150),Y(300),IS(300),IPPN(150),
     1LABEL(150,10),LBL(150,10),NT(20,25,25),LINE(100),TITLE(11,10),
     1NAME(10,10),W(50),WV(8,50),DEN(30),IPS(50),ISQ(50),PNU(30)
C
C             BLOCK A
C
C         READ DIVISION LINE SYMBOLS
      READ (5,49) BLANK,PLUS,DASH
   49 FORMAT (3A1)
      DO 45 J=1,100
   45 LINE(J)=DASH
      READ (5,54) ((TITLE(I,J),J=1,10),I=1,11)
   54 FORMAT (60A1/50A1)
      READ (5,55) (( NAME(I,J),J=1,10),I=1,10)
   55 FORMAT (60A1/40A1)
C.................PRINT THE TOPOLOGICAL FORMULAS
      PRINT 400
  400 FORMAT(1H1,20X,'FORMULAS'///)
      PRINT 401,(NAME(1,J),J=1,10),(TITLE(1,J),J=1,10),(LINE(J),J=1,45),
     1(NAME(2,J),J=1,10)
      PRINT 401,(NAME(2,J),J=1,10),(TITLE(2,J),J=1,10),(LINE(J),J=1,45),
     1(NAME(1,J),J=1,10)
  401 FORMAT(30X,10A1/5X,55A1/30X,10A1//)
      PRINT 402,((NAME(I,J),J=1,10),I=6,7),((NAME(I,J),J=1,10),I=4,5),
     1(TITLE(3,J),J=1,10),(LINE(J),J=1,45),(NAME(1,J),J=1,10)
  402 FORMAT(/15X,'1/2(',10A1,'+',10A1,'-',10A1,'-',10A1,')'/5X,
     1 55A1/30X,10A1//)
      PRINT 401,(NAME(3,J),J=1,10),(TITLE(4,J),J=1,10),(LINE(J),J=1,45),
     1(NAME(1,J),J=1,10)
      PRINT 401,(NAME(3,J),J=1,10),(TITLE(5,J),J=1,10),(LINF(J),J=1,45),
     1(NAME(8,J),J=1,10)
      PRINT 402,((NAME(I,J),J=1,10),I=4,7),(TITLE(6,J),J=1,10),
     1(LINE(J),J=1,45),(NAME(8,J),J=1,10)
      PRINT 401,(NAME(2,J),J=1,10),(TITLE(7,J),J=1,10),(LINE(J),J=1,45),
     1(NAME(8,J),J=1,10)
      PRINT 402,((NAME(I,J),J=1,10),I=6,7),((NAME(I,J),J=1,10),I=4,5),
     1(TITLE(8,J),J=1,10),(LINE(J),J=1,45),(NAME(2,J),J=1,10)
      PRINT 402,((NAME(I,J),J=1,10),I=5,7),((NAME(I,J),J=1,10),I=4,5),
     1(TITLE(9,J),J=1,10),(LINE(J),J=1,45),(NAME(3,J),J=1,10)
      PRINT 402,((NAME(I,J),J=1,10),I=4,7),(TITLE(10,J),J=1,10),
     1(LINE(J),J=1,45),(NAME(3,J),J=1,10)
      PRINT 402,((NAME(I,J),J=1,10),I=4,7),(TITLE(11,J),J=1,10),
     1(LINE(J),J=1,45),(NAME(2,J),J=1,10)
C
C                BLOCK B
C
 1100 CL=ITIME(0)*.01
      READ (5,1,END=1111) N,MB,IGR,IPORT
    1 FORMAT (4I5)
      PRINT  59
   59 FORMAT (1H1,20X,'INPUT DATA'///)
      PRINT 7, N,MB,IGR,IPORT
    7 FORMAT(/10X,'NODE=',I2,5X,'EDGES=',I2,5X,'GR CONDITION=',I2,',',I2,
     1'-PORT'//,
     1'  /EDGE NO/NODE NO/LABEL IN NK/NO BTN NODES/ADMITT/S-DGRE/'/,
     1'     LE      NX NY  NK(NY-1,NX)=K  IPP(K)      Y(LE)  IS(LE)')
      ITERM=2*N-1
      NV=N-1
      DO 2 I=1,NV
      DO 2 J=1,I
    2 NK(I,J)=0
      DO 6 I=1,MB
      READ (5,3)  LE,NX,NY,K,IPP(K),KJ,Y(LE),IS(LE)
    3 FORMAT(I10,2I3,3I5,F10.4,I5)
      LABEL(K,KJ)=LE
      LBL(K,KJ)=LE
      IPPN(K)=IPP(K)
      NK(NY-1,NX)=K
      PRINT 4,LE,NX,NY,NK(NY-1,NX),IPP(K),Y(LE),IS(LE)
    4 FORMAT (2I6,I3,2I10,F14.4,I5)
    6 CONTINUE
```

```
      PRINT 29
   29 FORMAT (//15X,'ORIGINAL T-TRIANGLE'/)
      DO 26 I=1,NV
      DO 26 J=1,I
   26 NT(1,I,J)=NK(I,J)
      DO 30 I =1,NV
   30 PRINT 31, (NT(1,I,J),J=1,I)
   31 FORMAT(/15X,25I3)
C
C             BLOCK C
C
      N3=3
      N4=4
      IF (IGR.EQ.1) N4=3
      MF=8
      IF (IPORT.EQ.1) MF=2
C
C             BLOCK D
C
      DO 10 L=1,MF
      GO TO (11,12,13,14,15,16,17,18),L
   11 NI=N
      GO TO 100
C.................TO YIELD THE NET FOR W1,1'
   12 IK=N3-1
      JK=1
      GO TO 200
C.................TO YIELD THE NET FOR W2,2'
   13 IK=N4-1
      JK=2
      GO TO 200
C.................TO YIELD THE NET FOR W1,2
   14 IK=1
      JK=1
      GO TO 200
C.................TO YIELD THE NET FOR W1',2'
   15 IK=N4-1
      JK=3
      IF(IK.LT.JK) GO TO 41
      GO TO 200
   41 PRINT 44,L,IK,JK
   44 FORMAT (///20X,' L=',I2,5X,'IK=',I2,'   JK=',I2//)
      DO 42 I=1,ITERM
   42 WV(L,I)=0.
      PRINT 43,(WV(L,J),J=1,ITERM)
   43 FORMAT (/10X,'NO TREE'//'  WV(L,J)=',15F6.2)
      GO TO 10
C..................TO YIELD THE NET FOR W1,2'
   16 IK=N4-1
      JK=1
      GO TO 200
C..................TO YIELD THE NET FOR W2,1'
   17 IK=N3-1
      JK=2
      GO TO 200
C..................TO YIELD THE NET FOR U--SHORT 2,2' AND1,1'
   18 IK=N3-1
      JK=1
  200 CALL       MODNET(IK,JK,N,L,NK,NT,NI,IPP,Y,IS,IPPN,LABEL,LBL,N4)
C
C             BLOCK E
C
  100 NV=NI-1
      DO 65 I=1,ITERM
   65 W(I)=0.
      PRINT 64,(NAME(L,J),J=1,10)
   64 FORMAT (///5X,10A1/)
      CALL       TREE(NI,NT,Y,IS,IPPN,W,N,LBL)
      DO 66 I=1,ITERM
   66 WV(L,I)=W(I)
      PRINT 68,(WV(L,I),I=1,ITERM)
   68 FORMAT (//10X,'WV(L,I)=',20(1X,F5.2))
   10 CONTINUE
C
C             BLOCK F
C
      PRINT 67
   67 FORMAT (1H1,20X,'PRINT OUT THE NETWORK FUNCTIONS'///)
C..................DRIVING POINT ADMITTANCE
      DO 79 I=1,ITERM
      DEN(I)=WV(2,I)
   79 PNU(I)=WV(1,I)
      LK=1
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C         COMPUTE THE OPEN-CIRCUIT IMPEDANCE,ZOC
C..................Z11
      DO 80 I=1,ITERM
      DEN(I)=WV(1,I)
   80 PNU(I)=WV(2,I)
      LK=2
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
      IF (L.EQ.2) GO TO 1000
C..................Z12,Z21
      DO 81 I=1,ITERM
      DEN(I)=WV(1,I)
   81 PNU(I)=0.5*(WV(6,I)+WV(7,I)-WV(4,I)-WV(5,I))
      LK=3
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C..................Z22
      DO 82 I=1,ITERM
      DEN(I)=WV(1,I)
   82 PNU(I)=WV(3,I)
      LK=4
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C         COMPUTE THE SHORT-CIRCUIT ADMITTANCE,YSC
C..................Y11
      DO 83 I=1,ITERM
      DEN(I)=WV(8,I)
   83 PNU(I)=WV(3,I)
      LK=5
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C..................Y12,Y21
      DO 84 I=1,ITERM
      DEN(I)=WV(8,I)
   84 PNU(I)=0.5*(WV(4,I)+WV(5,I)-WV(6,I)-WV(7,I))
      LK=6
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C..................Y22
      DO 85 I=1,ITERM
      DEN(I)=WV(8,I)
   85 PNU(I)=WV(2,I)
      LK=7
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C..................V2/V1
      DO 86 I=1,ITERM
      PNU(I)=0.5*(WV(6,I)+WV(7,I)-WV(4,I)-WV(5,I))
   86 DEN(I)=WV(2,I)
      LK=8
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C..................V1/V2
      DO 87 I=1,ITERM
      PNU(I)=0.5*(WV(6,I)+WV(7,I)-WV(4,I)-WV(5,I))
   87 DEN(I)=WV(3,I)
      LK=9
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
```

```
C.......................I2/I1
      DO 88 I=1,ITERM
      PNU(I)=0.5*(WV(4,I)+WV(5,I)-WV(6,I)-WV(7,I))
   88 DEN(I)=WV(3,I)
      LK=10
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
C.......................I1/I2
      DO 89 I=1,ITERM
      PNU(I)=0.5*(WV(4,I)+WV(5,I)-WV(6,I)-WV(7,I))
   89 DEN(I)=WV(2,I)
      LK=11
      CALL       RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
 1000 CL=ITIME(0)*.01-CL
C
C             BLOCK G
C
      WRITE(6,300) CL
  300 FORMAT (//5X,' TIME= ',F7.2,' SECONDS')
      GO TO 1100
 1111 STOP
      END




      SUBROUTINE MODNET(IK,JK,N,L,NK,NT,NI,IPP,Y,IS,IPPN,LABEL,LBL,N4)
      DIMENSION NK(20,25),IPP(150),Y(300),IS(300),IPPN(150),
     1LABEL(150,10),LBL(150,10),NT(20,25,25)
C         TRANSFER NK TO NT
C                 INPUT T-TRIANGLE
      NV=N-1
      DO 80 I=1,NV
      DO 80 J=1,I
   80 NT(1,I,J)=NK(I,J)
      PRINT 82,L,IK,JK
   82 FORMAT (///20X,' L=',I2,5X,'IK=',I2,'    JK=',I2//)
C         CHANGE COLUMN JK TO 1
  100 GO TO (1,2,3),JK
    3 ITEMP=NT(1,1,1)
      NT(1,1,1)=NT(1,2,2)
      NT(1,2,2)=ITEMP
    2 NW=NV-JK
      DO 5 I=JK,NV
      ITEMP=NT(1,I,1)
      NT(1,I,1)=NT(1,I,JK)
5     NT(1,I,JK)=ITEMP
    1 NX=IK-2
C         CHANGE ROW IK TO 1
      IF (IK-2)  10,7,6
    6 DO 8 I=1,NX
      ITEMP=NT(1,I+1,2)
      NT(1,I+1,2)=NT(1,IK,I+2)
    8 NT(1,IK,I+2)=ITEMP
    7 ITEMP=NT(1,1,1)
      NT(1,1,1)=NT(1,IK,1)
      NT(1,IK,1)=ITEMP
      NR=NV-IK
      IF (NR.EQ.0) GO TO 10
      DO 9 I=1,NR
      ITEMP=NT(1,IK+I,2)
      NT(1,IK+I,2)=NT(1,IK+I,IK+1)
    9 NT(1,IK+I,IK+1)=ITEMP
   10 DO 30 I=2,NV
      I1=NT(1,I,1)
      I2=NT(1,I,2)
      K=I1
      IF (I1.EQ.0) K=I2
      IF (K.EQ.0) GO TO 30
      JB=0
      IF (I1.EQ.0) GO TO 25
      JB=IPP(I1)
      DO 26 J=1,JB
   26 LBL(K,J)=LABEL(I1,J)
   25 IF (I2.EQ.0) GO TO 29
      JA=IPP(I2)
      DO 27 J=1,JA
      JB=1+JB
   27 LBL(K,JB)=LABEL(I2,J)
   29 IPPN(K)=JB
   30 NT(1,I-1,1)=K
      IF (NV.LT.3) GO TO 45
      DO 34 I=3,NV
      DO 34 J=3,I
      IP=NT(1,I,J)
      NT(1,I-1,J-1)=IP
      IF (IP.EQ.0) GO TO 34
      JP=IPP(IP)
      IPPN(IP)=JP
      DO 36 II=1,JP
   36 LBL(IP,II)=LABEL(IP,II)
   34 CONTINUE
   45 NI=NV
      NV=NI-1
C             NEW T-TRIANGLE CAN BE PRINTED HERE
      IF (L.EQ.8) GO TO 49
      GO TO 40
   49 DO 50 I =1,NV
      DO 50 J =1,I
      NK(I,J)=NT(1,I,J)
      IF (NK(I,J).EQ.0) GO TO 50
      K=NK(I,J)
      IPP(K)=IPPN(K)
      NJ=IPP(K)
      DO 51 JJ=1,NJ
   51 LABEL(K,JJ)=LBL(K,JJ)
   50 CONTINUE
      L=9
      IF (N4.EQ.4) GO TO 55
      IK=1
      JK=1
      GO TO 56
  55  IK=2
      JK=2
   56 GO TO 100
   40 RETURN
      END



      SUBROUTINE TREE(NI,NT,Y,IS,IPPN,W,N,LBL)
      DIMENSION NT(20,25,25),IROW(25),ICOL(25),MTREE(25,25),KEY(25),
     1IND(25),LIM(25),IPPN(150),W(50),LTREE(20,60),LBL(150,10),
     1Y(300),IS(300)
      IF (NI.EQ.0) STOP
      NV=NI-1
      N1=NI-1
      N2=NI-2
      N3=NI-3
      N4=NI-4
      N5=NI-5
      N6=NI-6
      N7=NI-7
```

```
N8=NI-8
N9=NI-9
N10=NI-10
N11=NI-11
N12=N-12
N13=N-13
N14=N-14
NO=0
NSUM=0
L=1
K=1
DO 1 I1=1,N1
M=1
IX=I1
IND(M)=M
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 34
IF (M.EQ.NV) GO TO 200
DO 2 I2=1,N2
M=2
IX=I2
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 33
IF (M.EQ.NV) GO TO 200
DO 3 I3 =1,N3
M=3
IX=I3
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 32
IF (M.EQ.NV) GO TO 200
DO 4 I4 =1,N4
M=4
IX=I4
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 31
IF (M.EQ.NV) GO TO 200
DO 5 I5 =1,N5
M=5
IX=I5
```

```
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 30
IF (M.EQ.NV) GO TO 200
DO 6 I6 =1,N6
M=6
IX=I6
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 29
IF (M.EQ.NV) GO TO 200
DO 7 I7 =1,N7
M=7
IX=I7
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 28
IF (M.EQ.NV) GO TO 200
DO   8 I8 =1,N8
M=8
IX=I8
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 27
IF (M.EQ.NV) GO TO 200
DO   9 I9 =1,N9
M=9
IX=I9
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 26
IF (M.EQ.NV) GO TO 200
DO 10 I10=1,N10
M=10
IX=I10
IND(M)=M
IF (IX.EQ.1)  IND(M)=IND(M-1)
CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
IF (KEY(M).EQ.0) GO TO 25
IF (M.EQ.NV) GO TO 200
```

```
      DO 11 I11=1,N11
      M=11
      IX=I11
      IND(M)=M
      IF (IX.EQ.1)  IND(M)=IND(M-1)
      CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
      IF (KEY(M).EQ.0) GO TO 24
      IF (M.EQ.NV) GO TO 200
      DO 12 I12=1,N12
      M=12
      IX=I12
      IND(M)=M
      IF (IX.EQ.1)  IND(M)=IND(M-1)
      CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
      IF (KEY(M).EQ.0) GO TO 23
      IF (M.EQ.NV) GO TO 200
      DO 13 I13=1,N13
      M=13
      IX=I13
      IND(M)=M
      IF (IX.EQ.1)  IND(M)=IND(M-1)
      CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
      IF (KEY(M).EQ.0) GO TO 22
      IF (M.EQ.NV) GO TO 200
      DO 14 I14=1,N14
      M=14
      IX=I14
      IND(M)=M
      IF (IX.EQ.1)  IND(M)=IND(M-1)
      CALL CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,LBL,LTREE)
  200 NO=NO+1
      NTR=1
      DO 15 I=1,NV
   15 NTR=NTR*LIM(I)
      NSUM=NSUM+NTR
C             PRINT T- MATRIX AT HERE
  180 CALL        TREEPR(NV,LTREE,Y,IS,LIM,W,N)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14),NV
   14 CONTINUE
   22 JX=14
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 23
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
   13 CONTINUE
   23 JX=13
```

```
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 24
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
   12 CONTINUE
   24 JX=12
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 25
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
   11 CONTINUE
   25 JX=11
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 26
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
   10 CONTINUE
   26 JX=10
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 27
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    9 CONTINUE
   27 JX=9
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 28
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    8 CONTINUE
   28 JX=8
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 29
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    7 CONTINUE
   29 JX=7
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 30
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    6 CONTINUE
   30 JX=6
      INX=IND(JX-1)
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 31
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    5 CONTINUE
   31 JX=5
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 32
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    4 CONTINUE
   32 JX=4
```

```
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 33
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    3 CONTINUE
   33 JX=3
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 34
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    2 CONTINUE
   34 JX=2
      IF (IROW(JX).EQ.0.OR.ICOL(JX).EQ.0) GO TO 35
      INX=IND(JX-1)
      CALL        NEWMTR(INX,JX,NI,NV,NT,K)
    1 CONTINUE
   35 CL=ITIME(0)*.01-CL
      PRINT  301,NSUM
  301 FORMAT(/15X,'TOTAL NO. OF TREES=',I7)
      RETURN
      END
```

```
      SUBROUTINE CONTL(M,NV,IX,NT,K,IROW,ICOL,MTREE,KEY,LIM,IPPN,
     1LBL,LTREE)
      DIMENSION NT(20,25,25),IROW(25),ICOL(25),MTREE(25,25),KEY(25),
     1LTREE(20,60),LIM(25),IPPN(150),LBL(150,10)
      MQ=M+1
      IROW(MQ)=0
      ICOL(MQ)=0
      IF ((M+IX).GT.NV) GO TO 12
      DO 11 J =MQ,NV
      ICOL(MQ)=NT(K,J,MQ)+ICOL(MQ)
      DO 11 I =1,M
   11 IROW(MQ)=NT(K,J,I)+IROW(MQ)
   12 JJ=0
      INC=0
      DO 16 I =1,M
      IF (NT(K,M,I).EQ.0) GO TO 16
      JJ=JJ+1
      MTREE(M,JJ)=NT(K,M,I)
      KF=NT(K,M,I)
      JP=IPPN(KF)
      DO 20 J=1,JP
      INC=INC+1
   20 LTREE(M,INC)=LBL(KF,J)
   16 CONTINUE
      LIM(M)=INC
      KEY(M)=JJ
      RETURN
      END


      SUBROUTINE NEWMTR(INX,JX,NI,NV,NT,K)
      DIMENSION  NT(20,25,25),ITEMP(25)
   19 NUM=JX-1
      K=JX-1
      DO 20 I =1,NUM
   20 ITEMP(I)=0
      NJ=NI-JX
      DO 21 I=JX,NV
      ITEMP(I)=NT(INX,I,JX)
   21 CONTINUE
C          MOVE THE ENTRIES ON BOTH SIDE OF OPT COLUMN JX UP ONE POSITION
C          THEN MOVE THE ENTRIES ON THE RIGHT OF JX ONE POSITION TO LEFT
      DO 22 I =1,NJ
      L=JX-2+I
      DO 24 J =1,L
      IF(J.GE.JX) GO TO 23
      NT(K,L,J)=NT(INX,L+1,J)
      GO TO 24
   23 NT(K,L,J)=NT(INX,L+1,J+1)
   24 CONTINUE
   22 CONTINUE
      DO 25 J =1,NV
   25 NT(K,NV,J)=ITEMP(J)
   35 RETURN
      END


      SUBROUTINE TREEPR(NV,LTREE,Y,IS,LIM,W,N)
      DIMENSION ITRL(25),IS(300),ITREE(25),LTREE(20,60),Y(300),W(50),
     1LIM(25),IDGR(25),YPRD(25)
```

```
M1=LIM(1)
DO   1 J1 =1,M1
L=1
KJ=LTREE(L,J1)
IDGR(L)=IS(KJ)
YPRD(L)=Y(KJ)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M2=LIM(L+1)
DO   2 J2 =1,M2
L=2
KJ=LTREE(L,J2)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M3=LIM(L+1)
DO   3 J3 =1,M3
L=3
KJ=LTREE(L,J3)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M4=LIM(L+1)
DO   4 J4 =1,M4
L=4
KJ=LTREE(L,J4)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M5=LIM(L+1)
DO   5 J5 =1,M5
L=5
KJ=LTREE(L,J5)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
IF (L.EQ.NV) GO TO 300
M6=LIM(L+1)
DO   6 J6 =1,M6
L=6
KJ=LTREE(L,J6)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
```

```
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M7=LIM(L+1)
DO  7 J7 =1,M7
L=7
KJ=LTREE(L,J7)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M8=LIM(L+1)
DO  8 J8 =1,M8
L=8
KJ=LTREE(L,J8)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M9=LIM(L+1)
DO  9 J9 =1,M9
L=9
KJ=LTREE(L,J9)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M10=LIM(L+1)
DO 10 J10=1,M10
L=10
KJ=LTREE(L,J10)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M11=LIM(L+1)
DO 11 J11=1,M11
L=11
KJ=LTREE(L,J11)
IDGR(L)=IS(KJ)+IDGR(L-1)
YPRD(L)=Y(KJ)*YPRD(L-1)
ITRL(L)=KJ
IF (L.EQ.NV) GO TO 300
M12=LIM(L+1)
DO 12 J12=1,M12
L=112
```

```
    KJ=LTREE(L,J12)
    IDGR(L)=IS(KJ)+IDGR(L-1)
    YPRD(L)=Y(KJ)*YPRD(L-1)
    ITRL(L)=KJ
    IF (L.EQ.NV) GO TO 300
    M13=LIM(L+1)
    DO 13 J13=1,M13
    L=113
    KJ=LTREE(L,J13)
    IDGR(L)=IS(KJ)+IDGR(L-1)
    YPRD(L)=Y(KJ)*YPRD(L-1)
    ITRL(L)=KJ
    IF (L.EQ.NV) GO TO 300
    M14=LIM(L+1)
    DO 14 J14=1,M14
    L=14
    KJ=LTREE(L,J14)
    IDGR(L)=IS(KJ)+IDGR(L-1)
    YPRD(L)=Y(KJ)*YPRD(L-1)
    ITRL(L)=KJ
300 PRINT 301,YPRD(L),IDGR(L),(ITRL(I),I=1,NV)
301 FORMAT (/5X,'TREE PROD(Y)=',F10.4,'S**',I2,5X,'TREE ',25I3)
    ISJ=IDGR(L)+N
    W(ISJ)=W(ISJ)+YPRD(L)
    GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14),NV
 14 CONTINUE
 13 CONTINUE
 12 CONTINUE
 11 CONTINUE
 10 CONTINUE
 9  CONTINUE
 8  CONTINUE
 7  CONTINUE
 6  CONTINUE
 5  CONTINUE
 4  CONTINUE
 3  CONTINUE
 2  CONTINUE
 1  CONTINUE
    RETURN
    END
```

```
      SUBROUTINE RESULT(DEN,PNU, N,LINE,ISP,ISQ,LK,TITLE,BLANK,PLUS)
      DIMENSION DEN(40),PNU(40),LINE(100),ISP(40),ISQ(40),TITLE(11,10),
     1 PSIGN(40),QSIGN(40)
      ITERM=2*N-1
      K=0
      I=0
      ESP=1.E-4
      DO 33 J=1,ITERM
      IF(ABS(DEN(J)).LT.ESP) GO TO 34
      K=K+1
      DEN(K)=DEN(J)
      ISQ(K)=J
      IF (K.EQ.1) GO TO 11
      IF (DEN(K)) 11,12,12
   11 QSIGN(K)=BLANK
      GO TO 34
   12 QSIGN(K)=PLUS
   34 IF(ABS(PNU(J)).LT.ESP) GO TO 33
      I=I+1
      PNU(I)=PNU(J)
      ISP(I)=J
      IF (I.EQ.1) GO TO 1
      IF (PNU(I)) 1,2,2
    1 PSIGN(I)=BLANK
      GO TO 33
    2 PSIGN(I)=PLUS
   33 CONTINUE
      IF (ISQ(1)-ISP(1)) 20,20,21
   20 M=ISQ(1)
      GO TO 25
   21 M=ISP(1)
   25 DO 36 J =1,K
   36 ISQ(J)=ISQ(J)-M
      DO 37 J=1,I
   37 ISP(J)=ISP(J)-M
      PRINT 72,((PSIGN(J),PNU(J),ISP(J)),J=1,I)
      PRINT 73,(TITLE(LK,J),J=1,10),(LINE(I),I=1,60)
      PRINT 74,((QSIGN(J),DEN(J),ISQ(J)),J=1,K)
   72 FORMAT(//20X,10(A1,F6.2,'S**'I2)/(20X,10(A1,F6.2,'S**'I2)))
   73 FORMAT (10X,120A1)
   74 FORMAT  (20X,10(A1,F6.2,'S**'I2)/(20X,10(A1,F6.2,'S**'I2)))
      RETURN
      END
```

FORMULAS

$$\text{DRYG } Y = \frac{V(Y)}{W1,1'(Y)}$$

$$Z11 = \frac{W1,1'(Y)}{V(Y)}$$

$$Z12 = Z21 = \frac{1/2(\ W1,2'(Y) + W2,1'(Y) - W1,2(Y) - W1',2'(Y))}{V(Y)}$$

$$Z22 = \frac{W2,2'(Y)}{V(Y)}$$

$$Y11 = \frac{W2,2'(Y)}{U(Y)}$$

$$Y12 = Y21 = \frac{1/2(\ W1,2(Y) + W1',2'(Y) - W1,2'(Y) - W2,1'(Y)\ )}{U(Y)}$$

$$Y22 = \frac{W1,1'(Y)}{U(Y)}$$

$$V2/V1 = \frac{1/2(\ W1,2'(Y) + W2,1'(Y) - W1,2(Y) - W1',2'(Y))}{W1,1'(Y)}$$

$$V1/V2 = \frac{1/2(\ W1,2'(Y) + W2,1'(Y) - W1,2(Y) - W1',2'(Y))}{W2,2'(Y)}$$

$$I2/I1 = \frac{1/2(\ W1,2(Y) + W1',2'(Y) - W1,2'(Y) - W2,1'(Y)\ )}{W2,2'(Y)}$$

$$I1/I2 = \frac{1/2(\ W1,2(Y) + W1',2'(Y) - W1,2'(Y) - W2,1'(Y)\ )}{W1,1'(Y)}$$

```
                INPUT DATA

         NODE = 4     EDGES= 6      GR CONDITION= 1, 1-PORT

/EDGE NO/NODE NO/LABEL IN NK/NO BTN NODES/ADMITT/S-DGRE/
  LE       NX NY   NK(NY-1,NX)=K   IPP(K)      Y(LE)   IS(LE)
  1         1  4          1           1        1.0000    -1
  2         3  4          2           1        2.0000     1
  3         2  4          3           2        1.0000     0
  4         2  4          3           2        3.0000     1
  5         2  3          4           2        1.0000    -1
  6         2  3          4           2        2.0000     0

             ORIGINAL  T-TRIANGLE

               0

               0  4

               1  3  2

   V(Y)

  TREE  PROD(Y)=     2.0000S** 0     TREE    1  3  2
  TREE  PROD(Y)=     1.0000S**-2     TREE    1  3  5
  TREE  PROD(Y)=     2.0000S**-1     TREE    1  3  6
  TREE  PROD(Y)=     6.0000S** 1     TREE    1  4  2
  TREE  PROD(Y)=     3.0000S**-1     TREE    1  4  5
  TREE  PROD(Y)=     6.0000S** 0     TREE    1  4  6
  TREE  PROD(Y)=     2.0000S**-1     TREE    1  2  5
  TREE  PROD(Y)=     4.0000S** 0     TREE    1  2  6

             TOTAL NO. OF TREES=        8

        WV(L,I)=  0.0    1.00   7.00  12.00   6.00   0.0    0.0

                     L= 2      IK= 2     JK= 1

   W1,1'(Y)

   TREE  PROD(Y)=    1.0000S**-2     TREE    5  1
   TREE  PROD(Y)=    2.0000S** 0     TREE    5  2
   TREE  PROD(Y)=    1.0000S**-1     TREE    5  3
TREE PROD(Y)=   3.0000S** 0     TREE    5  4
TREE PROD(Y)=   2.0000S**-1     TREE    6  1
TREE PROD(Y)=   4.0000S** 1     TREE    6  2
TREE PROD(Y)=   2.0000S** 0     TREE    6  3
TREE PROD(Y)=   6.0000S** 1     TREE    6  4
TREE PROD(Y)=   1.0000S** 1     TREE    1  3
TREE PROD(Y)=   3.0000S** 0     TREE    1  4
TREE PROD(Y)=   2.0000S** 1     TREE    2  3
TREE PROD(Y)=   6.0000S** 2     TREE    2  4

          TOTAL NO. OF TREES=      12

     WV(L,I)=  0.0    1.00   4.00 10.00 12.00  6.00  0.0
```

```
                   PRINT OUT THE NETWORK FUNCTIONS

             1.00S** 0+  7.00S** 1+ 12.00S** 2+  6.00S** 3
DRYG Y= ------------------------------------------------------------------
             1.00S** 0+  4.00S** 1+ 10.00S** 2+ 12.00S** 3+  6.00S** 4
             1.00S** 0+  4.00S** 1+ 10.00S** 2+ 12.00S** 3+  6.00S** 4
Z11    = ------------------------------------------------------------------
             1.00S** 0+  7.00S** 1+ 12.00S** 2+  6.00S** 3

  TIME=      0.25 SECONDS
```

```
              INPUT DATA

        NODE = 4     EDGES= 5     GR CONDITION= 2, 2-PORT

 /EDGE NO/NODE NO/LABEL IN NK/NO BTN NODES/ADMITT/S-DGRE/
   LE      NX  NY   NK(NY-1,NX)=K   IPP(K)      Y(LE)   IS(LE)
   1       1   2         1            1        1.0000     1
   2       2   3         2            1        2.0000    -1
   3       1   4         3            1        1.0000     0
   4       2   4         4            1        1.0000     1
   5       3   4         5            1        1.0000     0

            ORIGINAL  T-TRIANGLE

               1

               0   2

               3   4   5

     V(Y)

   TREE  PROD(Y)=     2.0000S** 0     TREE    1   2   3
   TREE  PROD(Y)=     2.0000S** 1     TREE    1   2   4
   TREE  PROD(Y)=     2.0000S** 0     TREE    1   2   5
   TREE  PROD(Y)=     1.0000S** 1     TREE    1   3   5
   TREE  PROD(Y)=     1.0000S** 2     TREE    1   4   5
   TREE  PROD(Y)=     1.0000S** 1     TREE    3   4   5
   TREE  PROD(Y)=     2.0000S** 0     TREE    3   4   2
   TREE  PROD(Y)=     2.0000S**-1     TREE    3   5   2

            TOTAL NO. OF TREES=       8

         WV(L,I)=  0.0    0.0    2.00   6.00   4.00   1.00   0.0

                     L= 2      IK= 2    JK= 1

     W1,1'(Y)

   TREE  PROD(Y)=     1.0000S** 1     TREE    1   3
   TREE  PROD(Y)=     1.0000S** 1     TREE    1   5
   TREE  PROD(Y)=     1.0000S** 2     TREE    1   4
   TREE  PROD(Y)=     2.0000S**-1     TREE    2   3
   TREE  PROD(Y)=     2.0000S**-1     TREE    2   5
```

```
TREE PROD(Y)=     2.0000S** 0     TREE   2  4
TREE PROD(Y)=     1.0000S** 1     TREE   3  4
TREE PROD(Y)=     1.0000S** 1     TREE   5  4

        TOTAL NO. OF TREES=        8

    WV(L,I)=  0.0    0.0    4.00   2.00   4.00   1.00   0.0

              L= 3      IK= 3    JK= 2

 W2,2'(Y)

TREE PROD(Y)=     2.0000S** 0     TREE   2  1
TREE PROD(Y)=     2.0000S**-1     TREE   2  3
TREE PROD(Y)=     1.0000S** 1     TREE   5  1
TREE PROD(Y)=     1.0000S** 0     TREE   5  3

        TOTAL NO. OF TREES=        4

    WV(L,I)=  0.0    0.0    2.00   3.00   1.00   0.0    0.0

              L= 4      IK= 1    JK= 1

 W1,2(Y)

TREE PROD(Y)=     2.0000S**-1     TREE   2  3
TREE PROD(Y)=     2.0000S** 0     TREE   2  4
TREE PROD(Y)=     2.0000S**-1     TREE   2  5
TREE PROD(Y)=     1.0000S** 0     TREE   3  5
TREE PROD(Y)=     1.0000S** 1     TREE   4  5

        TOTAL NO. OF TREES=        5

    WV(L,I)=  0.0    0.0    4.00   3.00   1.00   0.0    0.0

              L= 5      IK= 3    JK= 3

 W1',2'(Y)

TREE PROD(Y)=     2.0000S**-1     TREE   3  2
TREE PROD(Y)=     1.0000S** 1     TREE   3  4
TREE PROD(Y)=     1.0000S** 1     TREE   3  1
TREE PROD(Y)=     2.0000S** 0     TREE   2  1
TREE PROD(Y)=     1.0000S** 2     TREE   4  1
```

```
               TOTAL NO. OF TREES=      5

           WV(L,I)=  0.0    0.0    2.00   2.00   2.00   1.00   0.0

                         L= 6      IK= 3    JK= 1

    W1,2'(Y)

   TREE PROD(Y)=      1.0000S** 1      TREE    5  1
   TREE PROD(Y)=      1.0000S** 1      TREE    5  4
   TREE PROD(Y)=      2.0000S**-1      TREE    5  2
   TREE PROD(Y)=      2.0000S** 0      TREE    1  2
   TREE PROD(Y)=      2.0000S** 0      TREE    4  2

               TOTAL NO. OF TREES=      5

           WV(L,I)=  0.0    0.0    2.00   4.00   2.00   0.0    0.0

                         L= 7      IK= 2    JK= 2

    W2,1'(Y)

   TREE PROD(Y)=      1.0000S** 2      TREE    1  4
   TREE PROD(Y)=      1.0000S** 1      TREE    1  5
   TREE PROD(Y)=      1.0000S** 1      TREE    1  3
   TREE PROD(Y)=      1.0000S** 1      TREE    4  3
   TREE PROD(Y)=      1.0000S** 0      TREE    5  3

               TOTAL NO. OF TREES=     5

           WV(L,I)=  0.0    0.0    0.00   1.00   3.00   1.00   0.0

                         L= 8      IK= 2    JK= 1

    U(Y)

   TREE PROD(Y)=      1.0000S** 1      TREE    1
   TREE PROD(Y)=      2.0000S**-1      TREE    2
   TREE PROD(Y)=      1.0000S** 0      TREE    3
   TREE PROD(Y)=      1.0000S** 0      TREE    5

               TOTAL NO. OF TREES=      4

           WV(L,I)=  0.0    0.0    2.00   2.00   1.00   0.0    0.0
```

```
                 PRINT OUT THE NETWORK FUNCTIONS

           2.00S** 0+  6.00S** 1+  4.00S** 2+  1.00S** 3
DRYG Y= ----------------------------------------------------------
           4.00S** 0+  2.00S** 1+  4.00S** 2+  1.00S** 3
           4.00S** 0+  2.00S** 1+  4.00S** 2+  1.00S** 3
Z11    = ----------------------------------------------------------
           2.00S** 0+  6.00S** 1+  4.00S** 2+  1.00S** 3
          -2.00S** 0+  1.00S** 2
Z12=Z21= ----------------------------------------------------------
           2.00S** 0+  6.00S** 1+  4.00S** 2+  1.00S** 3
           2.00S** 0+  3.00S** 1+  1.00S** 2
Z22    = ----------------------------------------------------------
           2.00S** 0+  6.00S** 1+  4.00S** 2+  1.00S** 3
           2.00S** 0+  3.00S** 1+  1.00S** 2
Y11    = ----------------------------------------------------------
           2.00S** 0+  2.00S** 1+  1.00S** 2
           2.00S** 0  -1.00S** 2
Y12=Y21= ----------------------------------------------------------
           2.00S** 0+  2.00S** 1+  1.00S** 2
           4.00S** 0+  2.00S** 1+  4.00S** 2+  1.00S** 3
Y22    = ----------------------------------------------------------
           2.00S** 0+  2.00S** 1+  1.00S** 2
          -2.00S** 0+  1.00S** 2
V2/V1  = ----------------------------------------------------------
           4.00S** 0+  2.00S** 1+  4.00S** 2+  1.00S** 3
          -2.00S** 0+  1.00S** 2
V1/V2  = ----------------------------------------------------------
           2.00S** 0+  3.00S** 1+  1.00S** 2
           2.00S** 0  -1.00S** 2
I2/I1  = ----------------------------------------------------------
           2.00S** 0+  3.00S** 1+  1.00S** 2
```

```
            2.00S** 0  -1.00S** 2
I1/I2 = ----------------------------------------------------------
            4.00S** 0+  2.00S** 1+  4.00S** 2+  1.00S** 3

  TIME=     0.80 SECONDS
```

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library Naval Postgraduate School Monterey, California 93940	2
3.	Professor S. G. Chan Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	10
4.	LCDR W. T. Chang, CN 98 Jenchang Li, Yuchang, Tsaying Taiwan, Republic of China	2
5.	Commander, Naval Ordnance Systems Command Department of the Navy Washington, D. C. 20360	1

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE

Determination of Network Functions by Topological Method

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

CHANG, Wen-Tao

6. REPORT DATE June 1968	7a. TOTAL NO. OF PAGES 135	7b. NO. OF REFS 15
8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.		
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	
d.		

10. AVAILABILITY/LIMITATION NOTICES

This document is subject to special export controls and each transmittal to foreign nations may be made only with prior approval of the Naval Postgraduate School.

11. SUPPLEMENTARY NOTES N/A	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

Two new methods, suitable for either hand or digital computer calculations, are developed for the determination of all the trees of a nonoriented linear graph. Computer programs based on these methods are written. They greatly reduce the computation time while minimizing computer storage requirements.

In addition, a program for the determination of network functions is also developed by using topological relationships. This program is sufficiently generalized to permit its use for either a one port or a two port passive network without mutual inductances.

DD FORM 1 JAN 64 1473

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Topology; network functions of						
Network functions of topology						

DD FORM 1 NOV 65 1473 (BACK)
S/N 0101-807-6821

A-31409

102410

Thesis 102410
C372 Chang
c.2 Determination of
network functions by
topological methods.

14 JAN 86 33163

Th
C
3
c

Thesis 102410
C372
c.2 Chang
Determination of
network functions by
topological methods.

thesC372
Determination of network functions by to
3 2768 001 02371 6
DUDLEY KNOX LIBRARY